Naval Ocean Systems Center
San Diego, CA 92152-5000

Technical Document 1835
June 1990

Modeling the Environment of a Mobile Security Robot

H. R. Everett
Code 5303

G. A. Gilbreath
T. Tran
J. M. Nieusma
Code 535

NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152–5000

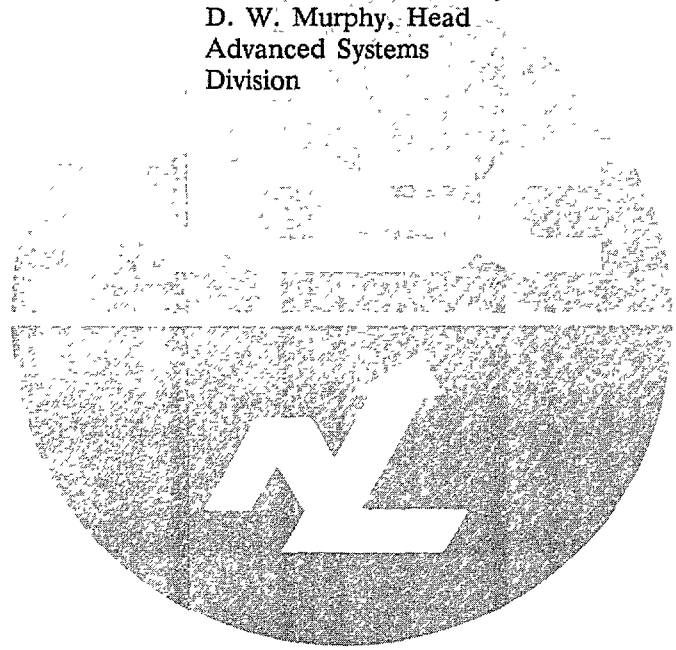
J. D. FONTANA, CAPT, USN
Commander

H. R. TALKINGTON, Acting
Technical Director

ADMINISTRATIVE INFORMATION

This document was prepared by Codes 5303 and 535 of the Naval Ocean Systems Center for the Director of Navy Laboratories.

Released under authority of
D. W. Murphy, Head
Advanced Systems
Division



DATE

JJ

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

FIGURES

CONTENTS (continued)

FIGURES (continued)

CONTENTS (continued)

FIGURES (continued)

CONTENTS (continued)

FIGURES (continued)

CONTENTS (continued)

FIGURES (continued)

CONTENTS (continued)

FIGURES (continued)

TABLES

CONTENTS (continued)

TABLES (continued)

ABSTRACT

ROBART II is a battery-powered autonomous robot being used by the Naval Ocean Systems Center (NOSC) in San Diego, CA as a testbed in research which seeks to provide a multisensor detection, verification, and intelligent assessment capability for a mobile security robot. The intent is to produce a robust automated system exhibiting a high probability of detection, with the ability to distinguish between actual and nuisance alarms, and capable of operation within areas already protected by fixed-installation motion detection sensors.

An architecture of 13 distributed microprocessors onboard the robot makes possible advanced control strategies and realtime data acquisition. Higher-level tasks (map generation, path planning, position estimation, obstacle avoidance, and statistical security assessment) are addressed by a Planner (currently a remote 80386-based desktop computer). Numerous sensors are incorporated into the system to yield appropriate information for use in position estimation, collision avoidance, navigational planning, and assessing terrain traversibility.

Although originally implemented as a purely autonomous vehicle with sophisticated world modeling capabilities, recent efforts have added an innovative form of remote (teleoperated) control to the system, allowing for manual direction of the robot's motion. This *reflexive teleoperation* frees the operator from the lower level concerns associated with direct teleoperation: speed of the vehicle and vehicle direction are servo-controlled by an onboard processor as a function of the surrounding environment in response to local sensor inputs, but under the high-level supervisory control of the remote operator.

The robot is also equipped with a multitude of sensors for environmental awareness in support of its role as an intelligent sentry. These sensors monitor both system and room temperature, relative humidity, barometric pressure, ambient light and noise levels, toxic gas, smoke, and fire. Intrusion detection is addressed through the use of five passive true-infrared body heat detectors, four passive optical motion detectors, ultrasonic motion detectors, microwave motion detectors, video motion detection, vibration monitoring, and discriminatory hearing. The realtime security assessment software computes a composite threat analysis by summing the weighted scores of alarmed sensors within a given zone. If the zone composite threat exceeds a dynamically computed threshold, a true alarm condition exists.

1.0 OBJECTIVE

The objective of this effort is to develop appropriate environmental awareness sensors and an environmental modeling capability for a mobile security robot, to support autonomous transit in a typical indoor scenario equipped with fixed-installation security sensors, with the ability to perform security detection and assessment operations both in stand-alone and supervised modes of operation.

2.0 BACKGROUND

2.1 PROJECT HISTORY

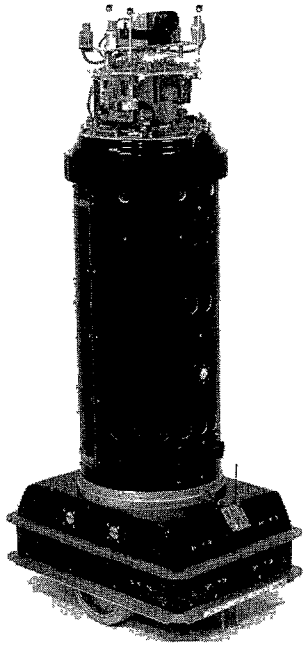


Figure 1. ROBART II, a mobile security robot, is employed as a research testbed at the Naval Ocean Systems Center, San Diego, CA.

Work on ROBART II (figure 1) was begun in October 1982, as a second-generation version of ROBART I (Everett, 1982), a prototype autonomous security robot (figure 2) developed in 1981 at the Naval Postgraduate School, Monterey, CA. The NOSC effort began in FY 87 as a 6.2 Independent Exploratory Development (IED) project entitled *Environmental Modeling for an Indoor Security Robot*, with focus on two critical technology base deficiencies identified by the Naval Sea Systems Command (NAVSEA) Office of Robotics and Autonomous Systems, then SEA 90G.

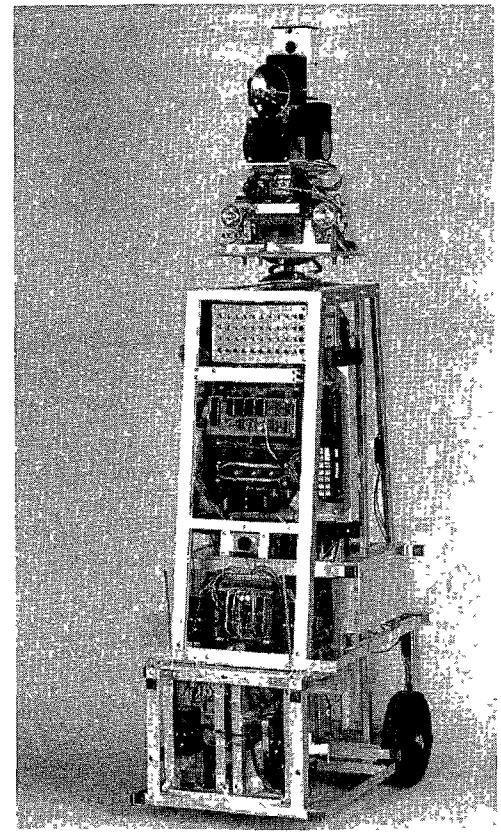


Figure 2. ROBART I, the original prototype security robot (1981-1982), was built at the Naval Postgraduate School in Monterey, CA.

The first of these addressed the navigational concerns which were hindering successful implementation of a number of NAVSEA robotic applications requiring mobility: the vehicle must be able to determine its position and orientation in the workspace, plan a path to its intended destination, and then execute that path without running into any obstructions. Numerous sensors, therefore, are incorporated into the system to yield appropriate information for use in map generation, position estimation, collision avoidance, navigational planning, and assessing terrain traversibility. An architecture of 13

microprocessors allows for robust realtime data acquisition and control.

The second need had been identified under previous and ongoing work in the 6.2 *Robotics for Physical Security Program* managed by the Defense Nuclear Agency. NOSC's intent in addressing this technology base deficiency was to produce a robust automated system exhibiting a high probability of detection with the ability to distinguish between actual and nuisance alarms. The robot is therefore also equipped with a multitude of sensors for environmental awareness in support of its role as an intelligent sentry. These sensors monitor both system and room temperature, relative humidity, barometric pressure, ambient light and noise levels, toxic gas, smoke, and fire. Intrusion detection is addressed through the use of five passive true-infrared body heat detectors, four passive optical motion detectors, a passive acoustical detection array, ultrasonic motion detectors, microwave motion detectors, video motion detection, vibration monitoring, and discriminatory hearing.

Success in these pursuits generated considerable outside interest, and in late FY 89 the program transitioned to 6.3 with complementary DoD funding from the Mobile Detection Assessment and Response System (MDARS) Program managed by the Army Armament Research, Development and Engineering Center (ARDEC) at Picatinny Arsenal, NJ. The scope of involvement was broadened to include enhancements to the modeling scheme to incorporate fixed installation security sensors (thereby allowing a mobile robot to operate in a secure area already protected by installed motion sensors) and inventory monitoring capability (allowing the robot to detect missing objects). In addition, the new MDARS requirement for some degree of teleoperation capability spawned the development of the highly acclaimed reflexive teleoperated control scheme, which frees the operator from the lower-level concerns associated with direct teleoperation; speed of the vehicle and vehicle direction are servo-controlled by an onboard processor in response to local sensor inputs, but under the high-level supervisory control of the remote operator.

As the effort moved closer towards actual production for preliminary field evaluation, additional emphasis was placed on automated self diagnostics. Approximately 256 internal circuitry checkpoints on the current testbed constantly monitor circuit performance, system configuration, operator controlled switch options, cable connections, and interface card integrity, etc., with speech output generated as appropriate to advise of any difficulties.

A second IED-funded project entitled *Distributed Robotic Control Architectures* was begun in FY 90 in support of this and other similar Navy applications. This effort seeks to develop a flexible distributed architecture with standardized interfaces for use on mobile robots, facilitating a modular approach at development (Laird, Smurlo, 1990). The modularity allows systems to be easily tailored to the actual needs of the application, with evolutionary upgrade potential. A modular third-generation robotic testbed (figure 3) is under development by Code 535 in conjunction with this work, with an identical system being developed in parallel by ARDEC personnel in a cooperative effort for MDARS evaluation.

2.2 SYSTEM OVERVIEW

Although ROBART II is significantly more complex than its predecessor, the intent is to explore practical solutions to some of the problems facing mobile autonomous robots; solutions that do not necessitate exorbitant amounts of computational resources, expensive in terms of acquisition costs, power consumption, size, and weight. Accordingly, the control architecture makes use of fairly simplistic microprocessors arranged in a distributed hierarchy so as to support the parallelism needed for the robot to be

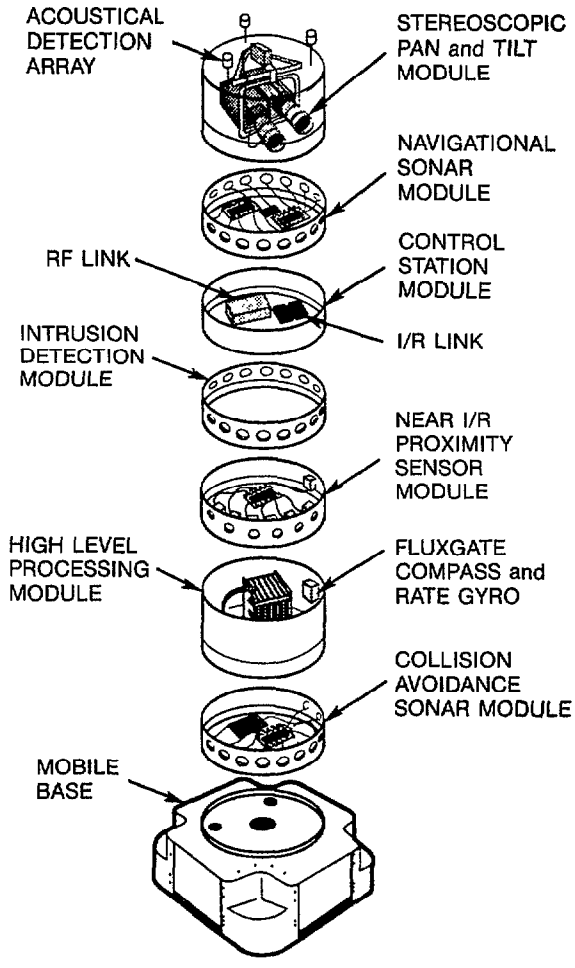


Figure 3. A modular third-generation testbed is now under construction at NOSC.

able to plan and execute activities in realtime. The most sophisticated of these microprocessors is a desktop 80386-based personal computer (Planner) which provides all the high-level planning and assessment functions.

The Planner currently communicates with the robot over a 1200 baud serial RF link. Thirteen microprocessors onboard the robot handle dedicated tasks such as ultrasonic ranging, head position control, acoustical and image processing, drive and steering, speech synthesis and recognition, and communications. Numerous sensors are incorporated into the system to yield appropriate information for use in position estimation, collision avoidance, navigational planning, and assessing terrain traversibility. The unit is housed in a rugged plastic and fiberglass body, and powered by a single rechargeable 12-volt, 32 amp-hour gel-cell battery.

2.2.1 The Navigational Problem

Any implementation involving a mobile robotic platform must deal with three fundamental issues related to navigation: (1) position estimation, (2) path planning, and, (3) collision avoidance. The robot must be able to determine current location and orientation, plan a route to the intended destination, and avoid impact with obstructions that may be encountered while in transit. Pursuit of these issues accounts for approximately sixty percent of the research effort in this project.

Accordingly, one of the first areas for concern in the evolution of a mobile robot design is the need to provide the system with sufficient environmental awareness to support intelligent movement. The first step towards this end consists of the acquisition of appropriate information regarding ranges and bearings to nearby objects, and the subsequent interpretation of that data.

The simplest case of navigational planning reduces the problem to two dimensions with a priori knowledge of the surroundings in the form of a memory map, or world model (figure 4). The task becomes one of trying to correlate a real-world sensor-generated image to the model, and extracting position and orientation. Several factors complicate the problem.

First of all, the real world is three-dimensional, and although the model represents each object as its projection on the X-Y plane, the robot's sensors will see things differently, as they are not physically located in a position to observe a plan view of the environment. This complicates the task of correlation, and the effects of various error sources can collectively impede a solution altogether. Secondly, the computational resources required are large, and the process is time consuming, requiring the robot at times to *stop and think*; the acquisition

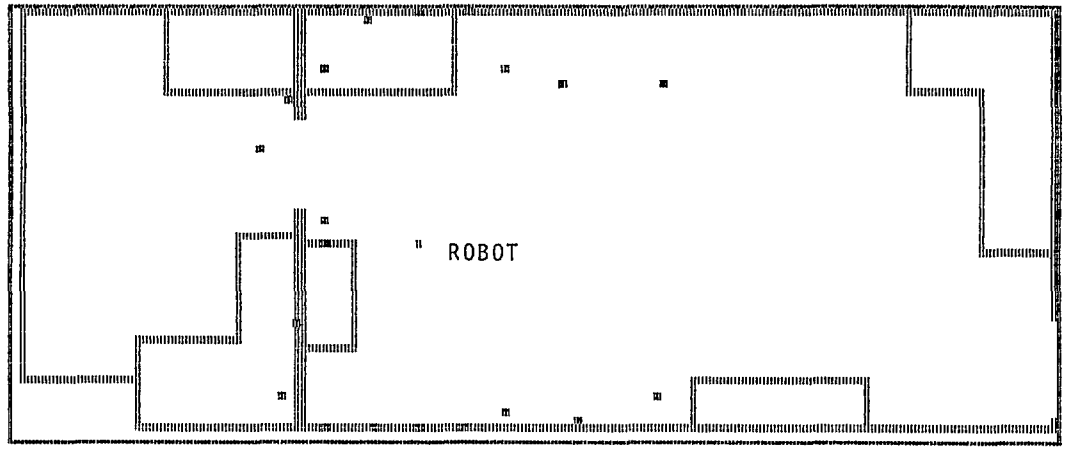


Figure 4. Ranges to nearby objects as seen by the navigational array of 24 ultrasonic sensors are marked on this X-Y planview of the robot's workspace.

of the data itself can take several seconds using ultrasonic ranging techniques, due to the relatively slow velocity of sound waves in air.

For purposes of path planning, objects are marked on the bit-mapped X-Y display by writing an appropriate value to the associated location in memory. No attempt is made during the modeling process to optimize through more efficient representation of free space, in that the research is oriented towards operation within relatively small (i.e., a building interior) bounded domains which can be quite congested. For the same reason, the path planning algorithms employ both brute force as well as A* search techniques, which are by no means the most efficient, but adequate for this application. Details of the actual mechanics of modeling the robot's surroundings are presented in section 4.2.

2.2.2 The Security Assessment Problem

One of the earliest perceived applications for such an autonomous device was acting as a sentry or security guard. Numerous sensors are readily available to support the detection functions (fire, smoke, intrusion, toxic gas, flooding, radiation, etc.) The ability to maintain an effective security presence under adverse (severe weather, degraded visibility) or even hazardous (nuclear, chemical, and biological) conditions is imperative, and appropriately addressed by mobile robotic systems of this type. Success, however, requires the needed research issues of environmental awareness, fixed and mobile sensor fusion, and intelligent security assessment be treated from a global systems integration point of view. Research towards this end is aimed at making the robot smart enough to distinguish between actual and nuisance alarms, and comprises the remaining forty percent of the effort.

Potential security functions assigned to a mobile sentry robot can be categorized into four general areas: (1) detection, (2) verification (3) assessment, and (4) response (Everett, 1988a). Detection is readily addressable by a multitude of commercially available sensors. Verification involves cross checking with other sensors to lessen the chances of a nuisance alarm, and depends heavily upon both the types of detectors employed and the operating environment. The assessment task acts upon the data collected to ascertain the nature of the disturbance, usually to determine if a response is necessary. The response itself must be tailored to the application, the operating scenario, and the nature of the situation.

From a security perspective, the environmental modeling problem is broken into two parts: (1) the local model, and (2) the global model. The local model deals only with those sensors physically located onboard the robot, and is implemented in the form of a series of adjoining pie-shaped *zones* converging upon the robot's immediate location. Section 5.2.1 discusses the manner in which a variety of intrusion detection sensors are arranged with overlaid fields-of-view within each of these zones. A synergistic effect is encountered when assessing the collective outputs of varying types of detectors, with a resulting increase in associated confidence levels. The necessary sensor integration need not require a complex system with extensive artificial intelligence, but can often be addressed through relatively low-level preprocessing of information.

The local model, however, is insufficient in itself to deal with the global picture: the robot is at times moving within an installation which is also guarded by a field of fixed sensors, and this same motion of the robot will trigger these fixed sensors, resulting in a constant series of nuisance alarms. In addition, the robot's onboard sensors must be correlated with these fixed sensors, as well as with those sensors of other robots. Accordingly, there is a need as well for a global model of the protected area, which will be discussed in section 5.2.2.

3.0 HARDWARE DESCRIPTION

3.1 COMPUTER ARCHITECTURE

At the top of the computer hierarchy is the Planner, a 16-bit machine which provides a home for the actual *intelligence* of the system (figure 5). This machine is currently an Intel 80386-based personal computer. The Planner performs the high-level tasks of map making, world model maintenance, path planning, obstacle avoidance, position estimation, sonar range plotting, and security assessment. In addition, the man-machine interface is implemented at this level. All remaining computers in the distributed hierarchy are physically located onboard the robotic vehicle.

CPU #1, the onboard system controller (Scheduler), is a single-board 6502-based system with off-board RAM and ROM expansion provided for a total of 64 kilobytes. Below this level in the hierarchy is another layer of seven CMOS 6502-based systems functioning as dedicated controllers. These microprocessors are interfaced to the Scheduler through an 8-bit parallel bus and a multiplexed RS-232 serial port, and assigned functions associated with communications, speech, drive and steering, ultrasonic ranging, acoustical processing, head motion, and vision. At the very bottom layer in the hierarchy are two more microprocessors, assigned the related responsibilities for speech synthesis and speech recognition.

CPU #2, the first of the microprocessors directly below the Scheduler, serves as a dedicated controller for positioning the head of the robot, which contains numerous directional sensors for use in navigation as well as intrusion detection. The head can rotate 100 degrees either side of centerline, and is driven by a 12-volt permanent magnet motor, with proportional pulse-width-modulation (PWM) control. Upon command from the Scheduler, CPU #2 can cause the head to assume a specified position, scan automatically for the robot's recharging station, or track the beacon as the robot moves toward it. Head position is derived from a sensing potentiometer and represented as an 8-bit number, with a resolution of 0.82 degrees per step.

CPU #3 performs all tasks associated with ultrasonic ranging. An 8-bit command from the Scheduler tells CPU #3 which sonars to fire; these units are sequentially enabled and the time required for each to detect an echo is converted to distance and stored. The sonar transducers themselves are interfaced to CPU #3 through three 12-input multiplexers. Upon completion of the full sequence, CPU #3 requests permission from the Scheduler to transmit the range information up the hierarchy, and when

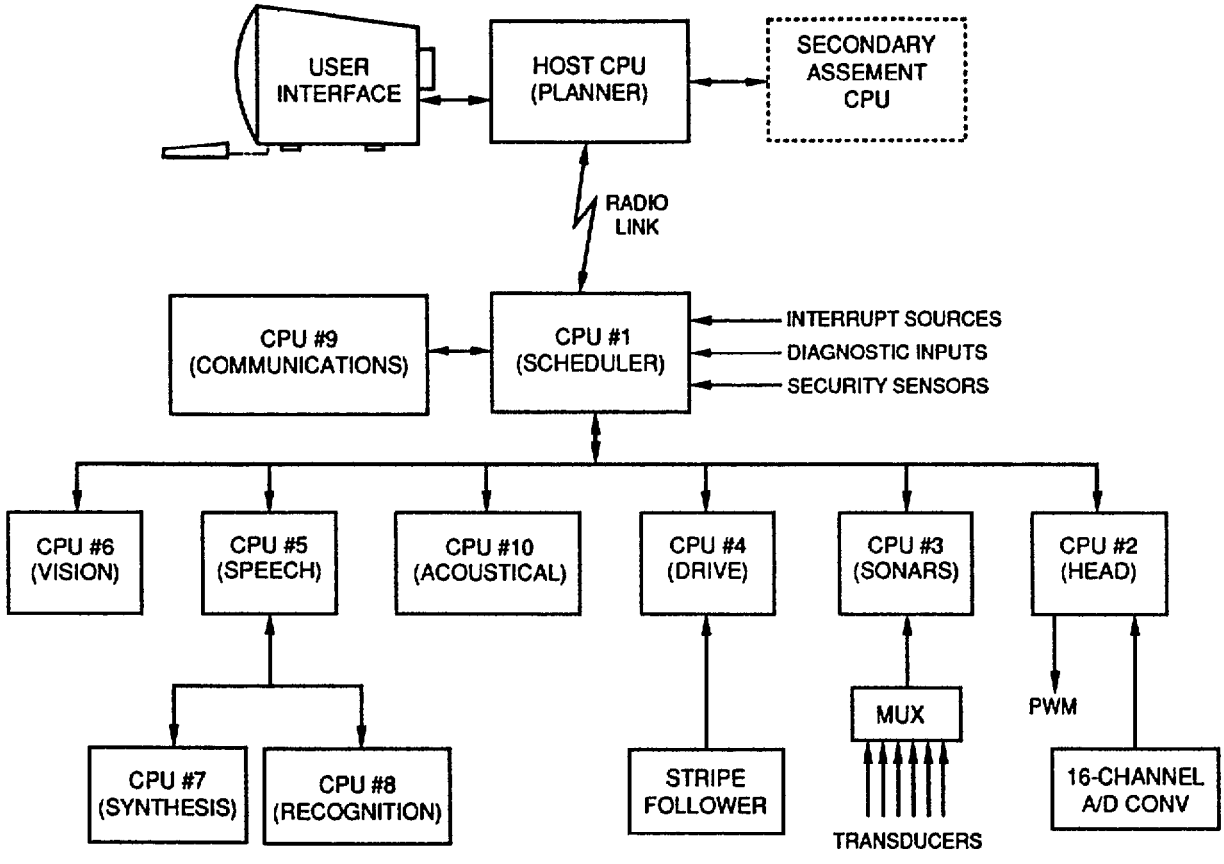


Figure 5. The distributed computer architecture for ROBART II employs a total of 13 microprocessors.

acknowledged passes it to the Scheduler. The sequence is repeated until altered upon receipt of a new command. The incoming ranges are read in and stored in the Scheduler's Page Zero for use by any software routines needing range information. Thus the software runs continuously in a repetitive fashion, sequentially activating the specified transducers, converting elapsed time to distance, correcting for temperature variations, and finally transmitting all range data at once to the Scheduler.

Another dedicated 6502-based controller handles all drive and steering control functions as directed by the Scheduler. Pulse-width-modulation is implemented through special hardware resident on an interface card connected between CPU #4 and the power transistors controlling the motors. For each motor there exists a directional control line and a modulated on/off control line. The width of the pulses appearing on these outputs is set by the commands from CPU #4 to the PWM interface card, and determines the speeds of the respective motors.

CPU #5 is assigned all functions related to speech input and output, and contains the system realtime clock. Below CPU #5 in the hierarchy are CPU #7, a DT1050 speech synthesizer, as well as CPU #6, a 6502-based speech recognition unit which can recognize up to 256 words. Another 6502-based system, CPU #8, is employed as a controller and signal processor for a line-oriented vision system, which is able to digitize and store any three horizontal lines of interest from a National Television Standards Committee (NTSC) composite video signal. CPU #9 is assigned all tasks associated with RF communications to and from the host, to include buffering and error checking of incoming and outgoing messages. CPU #10 functions as an

acoustical processor, fed by an array of three transducers, for use in intrusion detection.

Each of the above-mentioned processors and their associated systems will be treated in more detail in follow-on sections of the text. The entire electronics package, (figure 6) is easily removed by loosening two retaining screws, with all connections to the robot being made through numerous ribbon cables terminating in zero-insertion-force (ZIF) dual-inline-plug (DIP) sockets.

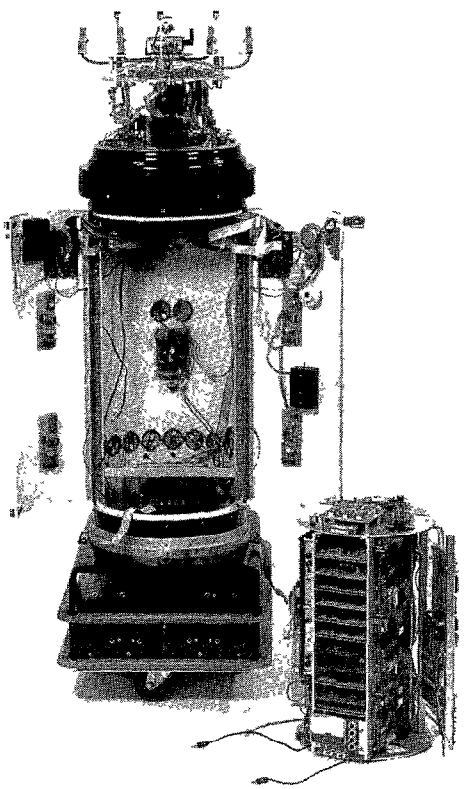


Figure 6. The onboard microprocessors and associated interface circuits are easily removed from the robot for servicing.

3.2 PROPULSION SYSTEM

The propulsion system uses two individually controlled drive wheels on either side of the base, with casters in front and rear for stability. This configuration allows the robot to spin in place about the vertical axis for maneuvering in congested areas. Conventional 8-inch wheelchair tires and motors provide a quiet, powerful propulsion system with minimal wheel slippage; decoupling clutches on each wheel allow for *free-wheeling* the robot around when the system is shutdown. The drive circuitry is based upon the RCA CDP-1878 variable duty cycle PWM controller chip, as shown in the block diagram of figure 7. The propulsion system can be deactivated manually for safety or other reasons by an observer using a small hand-held transmitter (similar to that used by a garage door opener). The upper body separates from the fiberglass base unit for portability and maintenance, and is held in place by four quick-release pins (figure 8).

The drive motors are independently controlled and synchronized by high resolution optical encoders attached to the armature shafts (figure 9). The phase-quadrature encoders produce 200 counts per turn of the motor armatures, which translates to 9725 counts per wheel revolution. Very precise displacement and velocity information is thus available for use in dead reckoning during maneuvering.

Robot displacement D along the path of travel is given by the equation

$$D = \frac{D_l + D_r}{2} \qquad (1)$$

where D_l = displacement of left wheel
D_r = displacement of right wheel

Similarly, the platform velocity V is given by the equation

$$V = \frac{V_l + V_r}{2} \qquad (2)$$

where V_l = displacement of left wheel
V_r = displacement of right wheel

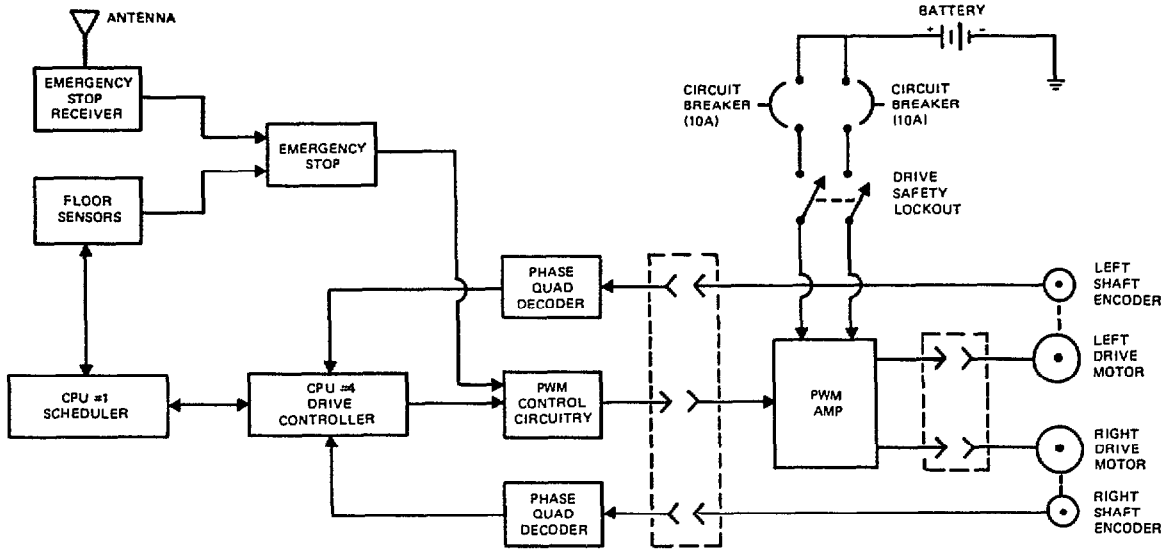


Figure 7. Block diagram of components associated with the robot's drive and steering system.

Referring to figure 10, arc D_l represents a portion of the circumference of a circle of radius d + b.

$$C_l = 2\ (d + b) \tag{3}$$

where C_l = Circumference of circle traced by left wheel
d = Distance between left and right drive wheels
b = Inner turn radius

In addition, the relationship

$$\frac{D_l}{C_l} = \frac{\theta}{2} \quad \text{yields} \quad C_l = \frac{2D_l}{\theta} \tag{4}$$

Combining equations 3 and 4 and solving for θ

$$\theta = \frac{D_l}{d + b} \tag{5}$$

Similarly, the shorter arc D_r represents a portion of the circumference of a circle of radius b.

$$C_r = 2\ b \tag{6}$$

where C_r = Circumference of circle traced by right wheel

And the relationship

$$\frac{D_r}{C_r} = \frac{\theta}{2} \quad \text{yields} \quad C_r = \frac{2\ D_r}{\theta} \tag{7}$$

Combining equations 6 and 7 and solving for b

$$b = \frac{D_r}{\theta} \tag{8}$$

Substituting this expression for b into equation 5

$$\theta = \frac{D_l}{d + \frac{D_r}{\theta}} = \frac{D_l - D_r}{d} \tag{9}$$

Note this expression for the change in vehicle orientation θ is a function of the displacements of the left and right drive wheels and is completely independent of the path taken.

Referring now to figure 11, wheel displacement D_l is given by the equation

$$D_l = R_{el}\ \phi \tag{10}$$

where R_{el} = effective left wheel radius
ϕ = wheel rotation (radians)

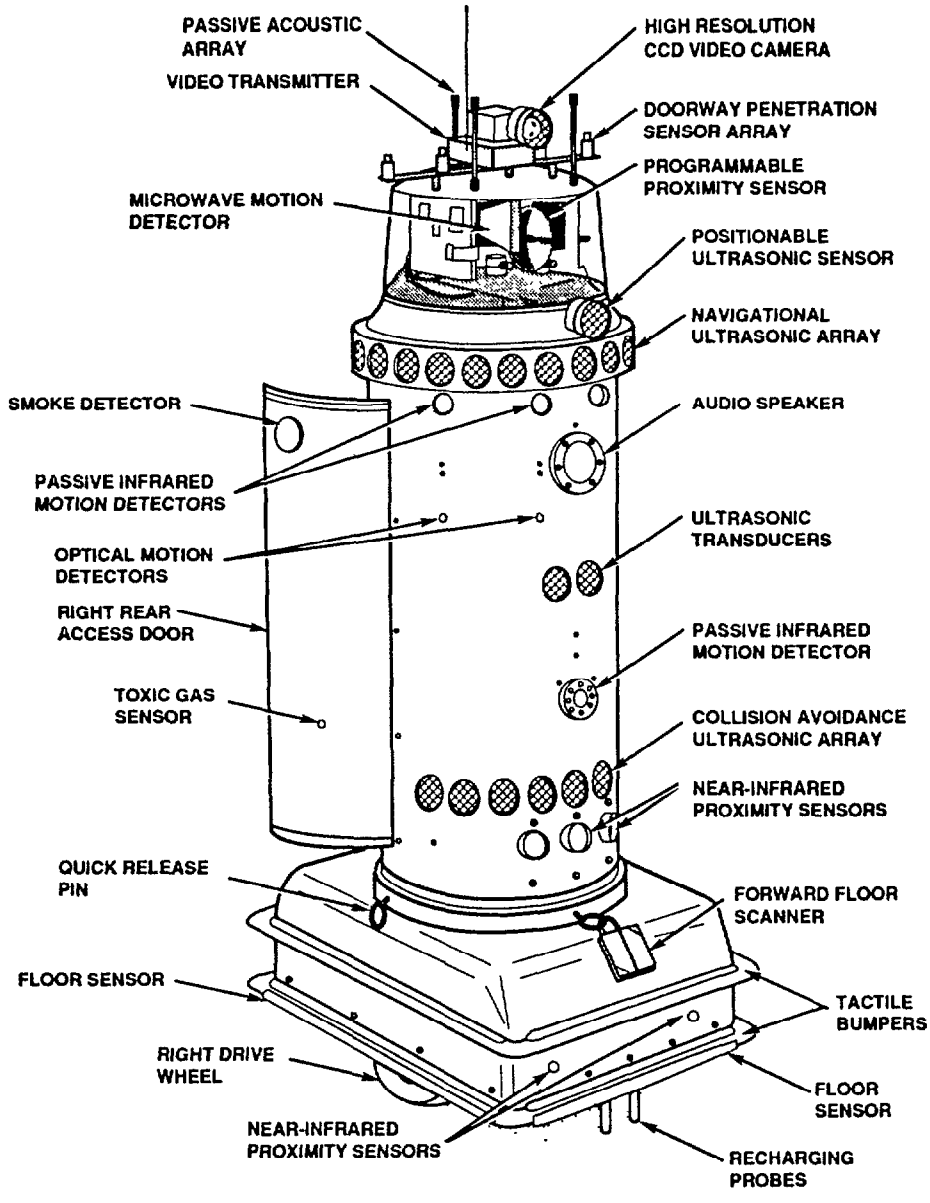


Figure 8. Physical configuration of some of the 130 external sensors employed on ROBART II to gather information for use in navigational and security assessment routines.

Expressing in terms of encoder counts yields

$$D_l = \frac{2\,N_l}{9,725}\,R_{el} \qquad (11)$$

where N_l = number of counts left encoder

Similarly, for the right drive wheel

$$D_r = \frac{2\,N_r}{9,725}\,R_{el} \qquad (12)$$

where N_l = number of counts right shaft encoder

R_{el} = effective right wheel radius

The drive controller will attempt to make the robot travel a straight line by ensuring N_r

and N_l are the same. Note, however, that effective wheel radius is a function of the compliance of the tire and the weight of the robot, and must be determined empirically. In actuality, R_{el} may not be equal to R_{er}, as in the case when several tires were tested on ROBART II in an attempt to obtain a matched set. For some tires, the compliance (and hence the effective radius) was found to actually vary as a function of wheel rotation ϕ. In any event, it was found to be virtually impossible to precisely match the left and right drive wheel displacements while motor velocities were held constant, due to minor variations in the tire materials. Slight deviations in the robot's heading were observed after straight line runs of 15 to 20 feet.

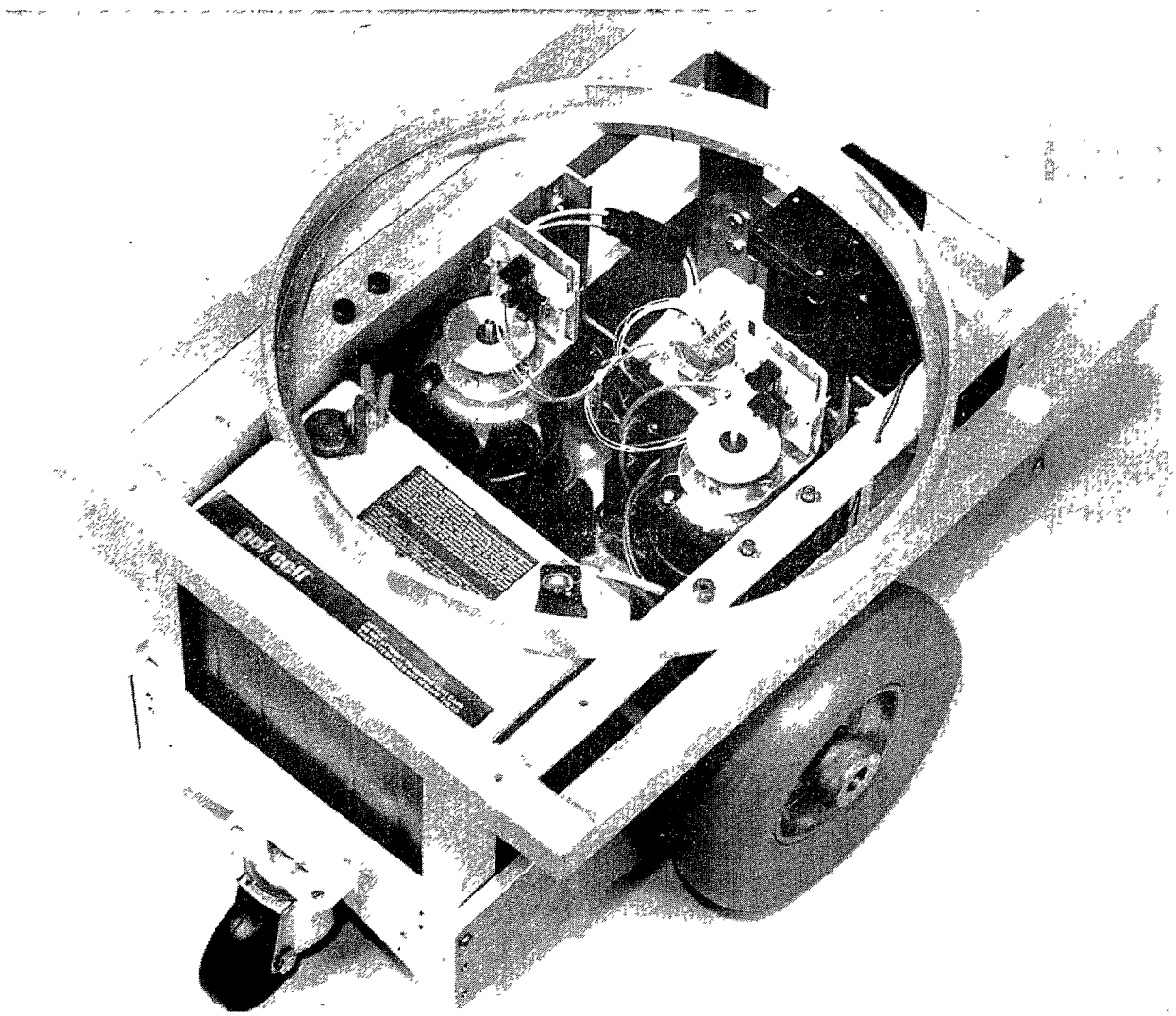


Figure 9. Phase quadrature optical encoders are attached to the armature shafts of the left and right drive motors for precise wheel displacement feedback.

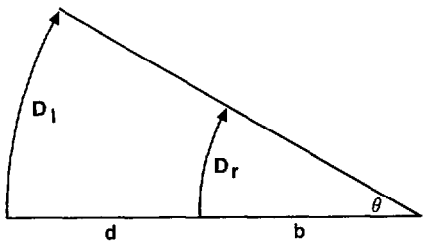


Figure 10. Arcs D_l and D_r represent portions of circles traced by the left and right wheels traveling at different but constant velocities, for change in robot heading θ.

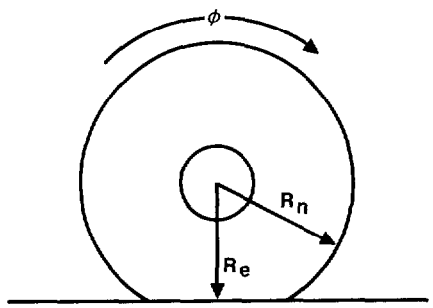


Figure 11. Effective wheel radius R_e is less than nominal wheel radius R_n, and a function of tire compliance.

Accordingly, to minimize dead reckoning errors, it was necessary to insert a *digital scaler* in series with the encoder output of the slower wheel (i.e., the smaller effective radius). The circuitry of figure 12 essentially blocks one encoder count for every X counts, X being a number between 239 and 255, determined by the setting of DIP switches 1 through 4. This *tricks* the drive controller into perceiving the motor speed to be slower than actual, and the software compensates accordingly by increasing the PWM cycle, speeding up the motor. The scaling factor can be experimentally adjusted for straight line travel.

Commands are passed by the Scheduler to the drive controller over a parallel bus as a series of hexadecimal bytes which specify (1) the direction in which to move or pivot, (2) the velocity, and, (3) the duration (distance or number of degrees).

Byte 1—Type Motion Requested (00 to 07)
- 00—Move forward
- 01—Move reverse
- 02—Pivot left
- 03—Pivot right
- 04—Offset left
- 05—Offset right
- 07—Stop

Byte 2—Requested Velocity (00 to FF)
- Upper nibble is the port motor velocity
- Lower nibble is the starboard motor velocity

Byte 3—Distance to Move in Inches (00 to FF) or,
Duration of Pivot in Degrees (00 to FF) or
Magnitude of Offset in Degrees/10

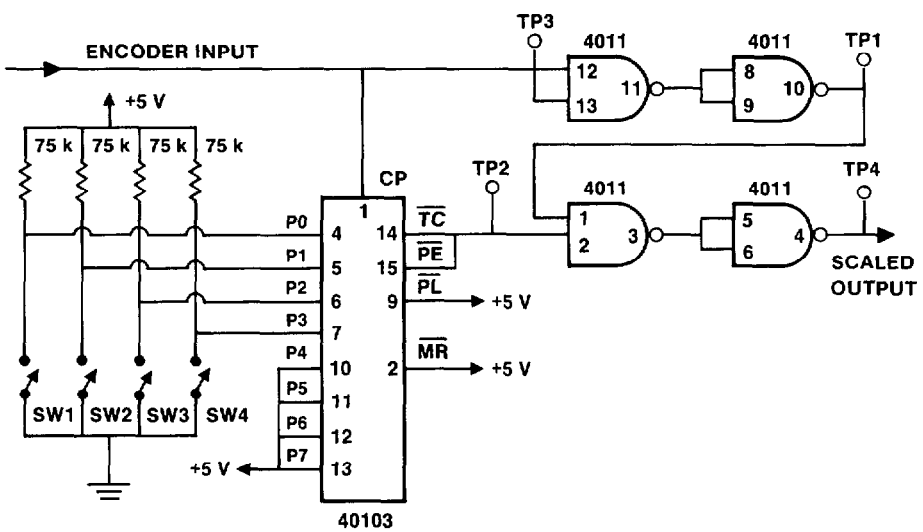


Figure 12. Schematic diagram of digital scaler used to compensate for difference in left and right drive wheel effective radius R_e.

(The special case of FF is interpreted for modes 00 to 03 as a continuous unbounded move or turn at the specified velocity.)

Velocity control and acceleration/deceleration ramping are performed by CPU #4 on an interrupt basis, while the main code performs all dead reckoning calculations. Cumulative X and Y components of displacement as well as current heading θ are passed up the hierarchy to the Scheduler at recurring intervals.

3.3 NAVIGATIONAL SENSOR SUBSYSTEMS

This section will briefly address the numerous sensors employed on ROBART II for navigation and collision avoidance purposes. Section 3.4 discusses security-related sensors.

3.3.1 Tactile Sensors

The only tactile sensors employed on ROBART II are two circumferential bumpers around the base unit. Each consists of a free-floating plastic strip encased in a fixed housing, spring loaded to be normally in the extended position. A series of microswitches is arranged behind these housings such that individual switch elements are engaged by any displacement of the strip. When the bumper comes in contact with another surface, the floating strip is locally depressed, and activates the appropriate microswitch to provide geometric resolution of the point of impact. This facilitates intelligent recovery by the collision avoidance software. The most significant component of this continuous bumper design is the corner piece (figure 13), designed with an angled cut at both ends to mate with the floating strips in the linear encasings. When a corner comes in contact with another surface, it will press against a floating strip, which activates the microswitch nearest the corner. The angled construction also permits lateral motion of the strips within their encasings when responding to oblique impacts. The configuration doubles as a protective bumper for the surface of the robot base as well as the near-infrared proximity sensors mounted thereon (section 3.3.3).

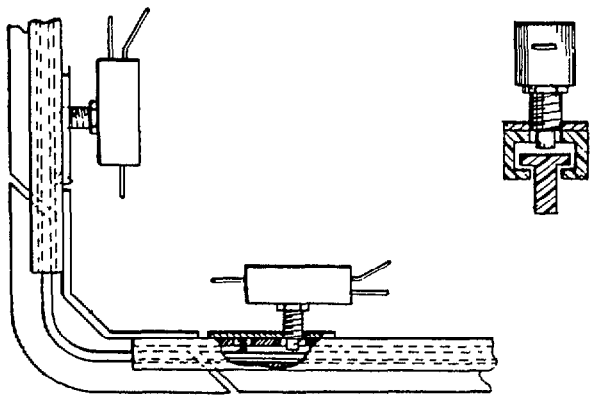


Figure 13. Spring-loaded tactile bumpers are designed to activate a series of microswitches when locally depressed so as to provide geometric resolution of the point of impact.

3.3.2 Ultrasonic Ranging

Ultrasonic ranging is today the most commonly used technique employed on indoor mobile robotic systems for obtaining range information describing the position of nearby objects, primarily due to the ready availability of low cost systems, and their ease of interface. Over the past decade, much research has been conducted investigating the applicability of ultrasonic ranging for purposes of map making, position estimation, collision avoidance, and motion detection.

The ranging modules employed on ROBART II were made by Texas Instruments for use with the Polaroid electrostatic transducer, and were selected due to their low cost, high reliability, and ease of interface. The Polaroid ranging module is an active time-of-flight device developed for automatic camera focusing, and determines the range to target by measuring elapsed time between the transmission of a 50-kHz ultrasonic waveform and the detected echo.

ROBART's ultrasonic ranging capability was upgraded in FY 87 from the original six transducers (Everett, 1985) to a total of 36 discrete sensors to better support the research

thrust of environmental modeling. For obstacle avoidance purposes, a fixed array of 11 transducers is installed on the front of the body trunk to provide distance information to objects in the path of the robot. A ring of 24 additional ranging sensors (15 degrees apart) is mounted just below the robot's head, and used to gather range information for position estimation. A final ranging unit is located on the rotating head assembly, allowing for distance measurements to be made in various directions.

The physical configuration of the ranging sensors (see again figure 8) is based upon a heuristic developed during work with an earlier prototype (Everett, 1982), which simply observes that for purposes of modeling a robot's surrounding environment, the taller an observed object, the more permanent its position is likely to be. In other words, smaller objects are likely to be more transitory in nature than larger objects. In an indoor scenario, typical examples of such moving entities might include trashcans, boxes on the floor, or chairs. Less subject to motion, but still not rigidly constrained, would be desks and file cabinets. At the stationary end of the spectrum are structural walls, which extend from floor to ceiling.

It would be desirable, therefore, to use as navigational aids those entities in the environment which are less susceptible to relocation over the period of time the robot is operating in their vicinity. This suggests locating the associated navigational sensors as high as possible on the robot's structure. Conversely, transient objects not likely to be represented in the world model are best detected by sensors located near the floor.

The 24 ultrasonic ranging units in the Navigational Array are interfaced to two Polaroid ranging modules through dual 12-channel multiplexers (figure 14). This way the microprocessor *sees* only two transducers at a time through their respective multiplexers, and the software merely executes in a loop, incrementing each time the index which enables a specific pair of transducers. A third multiplexer is employed to interface the transducers in the Collision Array to a final Polaroid module.

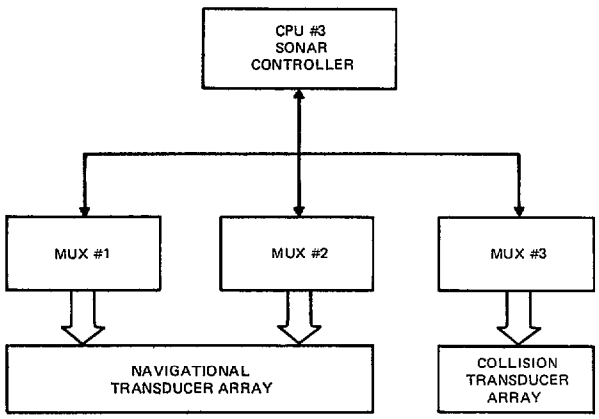


Figure 14. Block diagram of sonar multiplexing scheme which handles a total of 36 ultrasonic transducers.

The heart of each multiplexer is a 4067 analog switch operating in the digital mode (figure 15). To simplify the circuitry involved, all timing and time-to-distance conversions are done in software. Three control lines are involved in the interface of the Polaroid ultrasonic circuit board to a microprocessor. The first of these, referred to as VSW initiates operation when brought high to +5 volts. A second line labeled XLOG signals the start of pulse transmission, while the line labeled MFLOG indicates detection of the first echo. The controlling microprocessor must therefore send VSW high, monitor the state of XLOG and commence timing when transmission begins (approximately 5 milliseconds later), and then poll MFLOG until an echo is detected or sufficient time elapses to indicate there is no echo.

Four input/output (I/O) lines from CPU #3 handle the transducer enabling function, activating simultaneously the 4067 multiplexers for VSW, XLOG, and MFLOG. The binary number placed on these I/O lines by the microprocessor determines which channel is selected, all other channels assume a high impedance state. Three

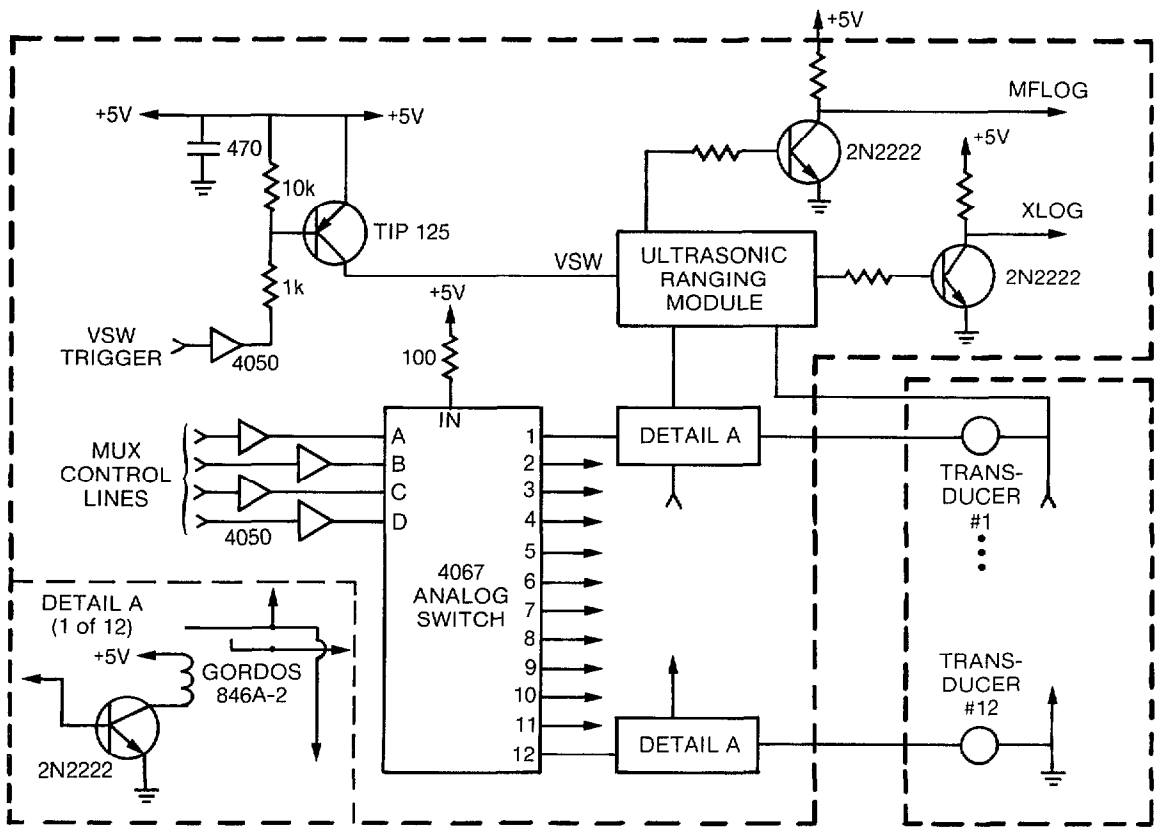


Figure 15. Schematic diagram of multiplexing unit which interfaces 12 transducers to a single Polaroid ranging module.

other I/O lines carry the logic inputs to the microprocessor from the multiplexers for XLOG and MFLOG, and from the microprocessor to the multiplexer for VSW. A final I/O line on the same port is used to power down the interface circuitry and the ranging units when not in use to save battery power.

A second parallel port is used to receive commands from the Scheduler which tell CPU #3 to power up the ranging units, and then which sensors to sequentially activate. Commands are in the form of an 8-bit binary number represented in hexadecimal format, where the upper nibble represents the starting ID and the lower nibble the ending ID for the sequence. For example, the command $16 would mean activate and take ranges using sensors one through six sequentially, whereas the command $44 would cause only sensor four in the array to be repeatedly activated. Each time through the loop upon completion of the sequence, the stored ranges are transmitted up the hierarchy to the Scheduler over an RS-232 serial link, with appropriate handshaking. The sequence is repeated in similar fashion until such time as the Scheduler sends a new command down, or advises CPU #3 to power down the ranging system with the special command $FF.

The software is structured as shown in figure 16. When energized by the Scheduler, CPU #3 does a power-on reset, initializes all ports and registers, and waits for a command. When a command is latched into the I/O port, a flag is set automatically, alerting the microprocessor which reads the command and determines the

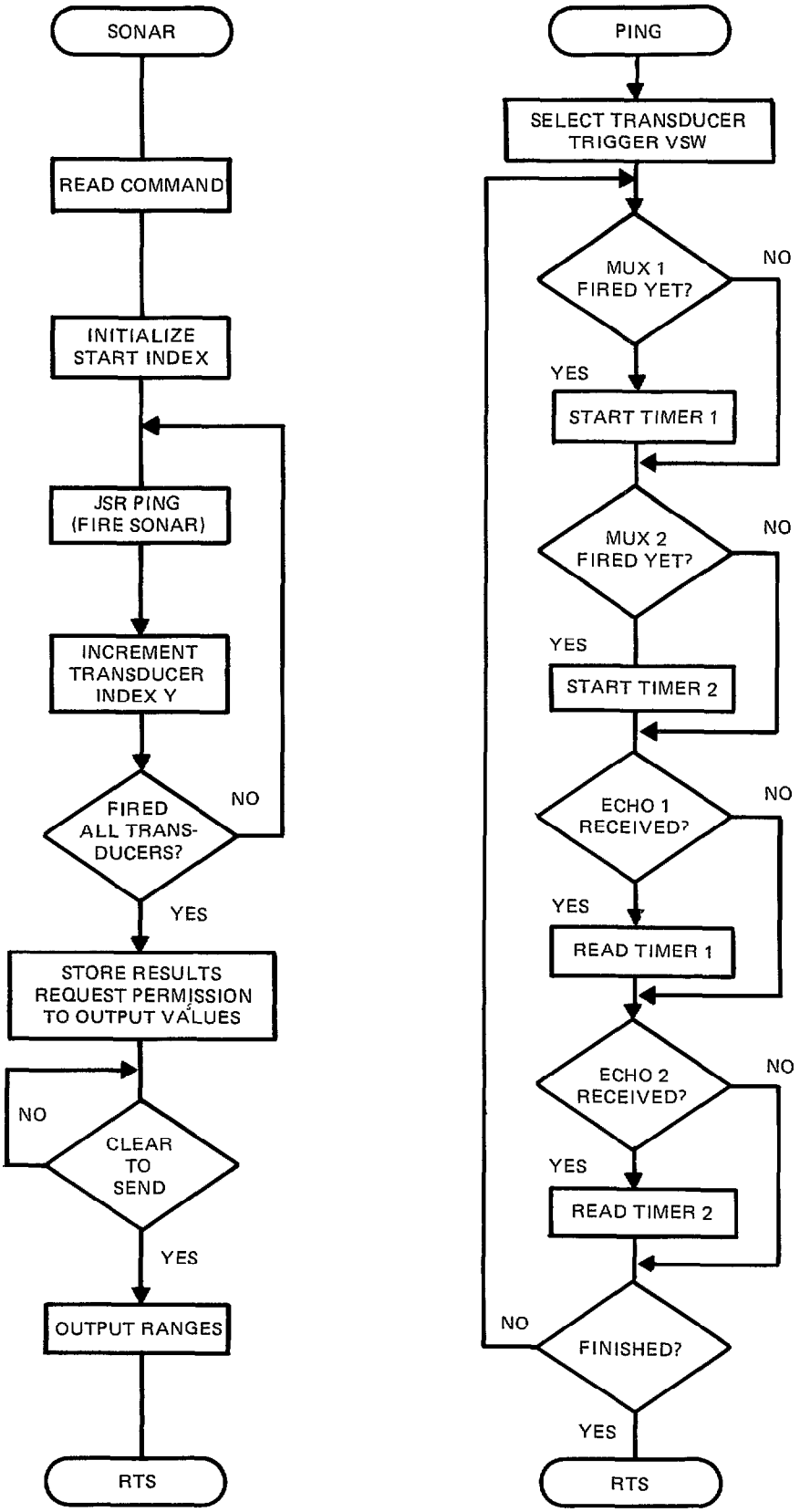


Figure 16. High level flowchart of the ultrasonic ranging software which runs on CPU #3.

starting and ending identities of the rangefinders to be sequentially activated. The interface circuitry and ranging units are powered up, and the Y Register is set to the value of the first transducer to be fired.

Subroutine PING is then called, which enables the particular channel of the multiplexer interface dictated by the contents of the Y Register. The VSW control line is sent high, which initiates operation of the ranging module with the selected transducer. The software watches the multiplexer output XLOG for indication of pulse transmission, before initiating the timing sequence. The contents of the timing counter, representing elapsed time, can be used to calculate range to the target. If this value ever exceeds the maximum specified range of the system, the software will exit the loop, otherwise the counter runs until MFLOG is observed to go high, indicating echo detection. Upon exit from the timing loop, the range value for that particular transducer is saved in indexed storage, and after a brief wait period (to let residual echoes decay), subroutine PING returns to the main program.

The Y Register is incremented to enable the next ranging module in the sequence, and subroutine PING is called again. This process is repeated until the Y Register equals the value of the ending index, signifying all transducers in the sequence specified by the Scheduler have been individually activated. CPU #3 then requests permission from the Scheduler to transmit all the stored range values via the RS-232. When acknowledged, the ranges are sequentially dumped out the serial interface and placed by the Scheduler in Page Zero indexed storage. Upon completion, CPU #3 checks to see if a new command has been sent down altering the ranging sequence, and repeats the process using the appropriate starting and ending indexes. Thus the software runs continuously in a repetitive fashion, sequentially activating the specified ranging modules, converting elapsed time to distance, storing the individual results, and then finally transmitting all range data at once to the Scheduler, which is thus freed from all associated overhead.

3.3.2.1 Temperature Compensation. The speed of sound in air is proportional to the square root of temperature in degrees Rankine, which for the temperature variations likely to be encountered in this application, results in a significant effect even considering the short ranges involved. Temperature variations over the span of 60° to 80°F can produce a range error as large as 7.8 inches at a distance of 35 feet (Everett, 1985). Fortunately, this situation is easily remedied through the use of a correction factor based upon the actual room temperature, available with an accuracy of 0.5°F from an external sensor mounted on the left access door. Current ambient temperature is passed to CPU #3 by the Scheduler, where the compensation calculation is performed during the brief waiting period between firing of the individual transducers in the array. The formula is simply: actual range equals measured range times the correction factor, where the correction factor is the square root of the ratio of actual temperature to standard temperature, in degrees Rankine.

3.3.2.2 Debugging the System. On 27 January 1990, a test was being run to determine the robot's dead-reckoning accuracy following a recalibration procedure necessitated by an increase in overall system weight. (As discussed in section 3.2, tire compliance causes a decrease in effective wheel radius as the applied weight is increased). For this test, the robot was directed by the path planner to execute a rectangular pattern 30 feet long by 8 feet wide, returning to the point of original departure. The distance between the final position and the starting position divided by the distance traveled was used as a relative indicator of accuracy. A downward-pointing helium-neon laser temporarily affixed to the base of the robot provided the necessary positional index.

Several repeats of the test were desired for statistical purposes. Initial runs around the test pattern were conducted without incident for about 30 minutes, when for some unexplained

reason, the robot began to stop as though confronted by an obstacle. The system then planned and executed a collision avoidance maneuver, which invalidated the test in that the total distance traversed was no longer held constant for each run. A quick examination of data from the robot's debug port revealed the phantom obstacle was being sensed by the Collision Avoidance Sonar Array and not the complementary near-infrared proximity sensors (figure 8).

Since this problem was not initially present, preliminary speculation was that it was somehow related to declining battery potential, in that the system voltage had fallen during the tests to a point below which the onboard supervisory computer normally initiates an automatic recharging sequence. With the drive wheel clutches disengaged to inhibit forward motion, the robot was connected to an umbilical providing auxiliary power, and erroneous sonar readings were periodically observed to be still present as battery voltage returned to normal.

At this point electrical noise within the robot's distribution system became the target of investigation, with the primary suspect being the differential drive motors and their associated PWM amplifiers. The drive safety switch was disabled, thus eliminating the possibility of motor noise, yet the problem persisted. No electrical noise whatsoever was observed at the input to the sonar system multiplexers or the controlling microprocessor, and the erroneous range readings were becoming more frequent. The bug was proving quite elusive.

Attention now shifted to an examination of the actual range data. In the execution of a planned path segment, only the center five ranging transducers in the Collision Avoidance Array are fired, thus providing sufficient forward coverage with a fast enough update rate to detect obstructions in the intended path of travel. The rightmost sensor was found to be periodically returning a range reading of approximately 4 to 11 inches, when in fact no targets were closer than 10 feet within its field of view. All other sensors appeared to be functioning normally.

This observation seemed to indicate something was wrong with this particular transducer, or its associated interface to the multiplexer, and so the robot was driven in a teleoperated fashion back into the adjoining lab area for investigation and repair. The large area required to execute the rectangular test pattern could only be accommodated in an open bay in the center of the building, but the robot had operated in this bay on many previous occasions without incident. Surprisingly, however, the problem could not be replicated in the lab, and so the robot was commanded to return to the bay.

Immediately upon entry into the open bay area, the problem reappeared, but this time a different sensor was affected. After examining the data closely, it was decided to allow the robot to explore the bay, while operators observed its actions for signs of faulty sonar readings. The rectangular bay in this particular building is surrounded on three sides by office and laboratory spaces (figure 17), with a large roll-up industrial garage door along the fourth side for exterior access. At the opposite end from the garage door, hallways extend to the left and right to additional exterior exits which are conventional 36-inch doorways. It soon became apparent that the sonar system behaved normally in the hallways, as it had done in the lab area, but became erratic in the open bay. Furthermore, it seemed that the closer the robot approached the garage door, the worse the problem became. The bug was now beginning to take on the identity of acoustical interference.

Numerous factors affect the operation of ultrasonic ranging systems operating in air, and are well documented in the literature (Everett, 1985). Interference of the type observed in this case can usually be attributed to some external source of ultrasonic noise, or peculiarities associated with the way the sonar beam interacts with target surfaces. In a sequentially fired array such as employed on ROBART II, there is always the danger of adjacent sensor interaction, where returning energy from one transducer is actually

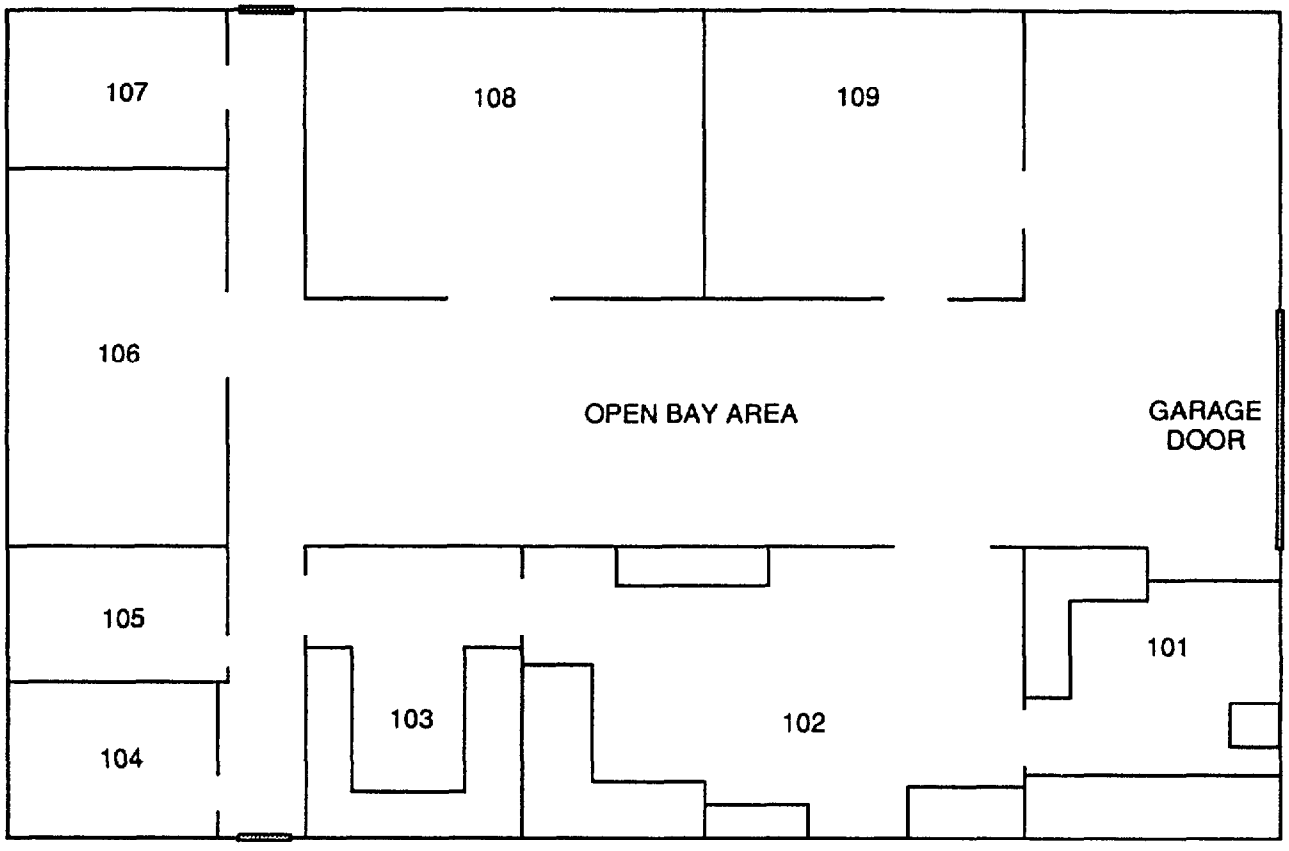


Figure 17. Map of Building F-36, showing location of garage door relative to open bay area where test was conducted.

received by the next transducer in the firing sequence, which happens to be listening at the time of arrival of the residual echo. This problem is minimized through proper selection of the firing order, and by virtue of the fact that the receiver ramps up the gain as a function of elapsed time, so that faint echoes from distant targets are usually too weak to register with a ranging unit that has just fired a new burst.

Nevertheless, residual echoes were immediately suspect when attention focused on the garage door; it was noted upon examination that the many corrugated folds employed in the roll-type construction of the door actually formed a highly cooperative target surface of retro-reflective strips. It was theorized that such a target could perhaps return more energy than a typical wall surface, thereby enabling detection of residual echoes by the next sensor to be fired. The second sensor would mistakenly identify the echo as its own, and calculate an erroneous range. This would account for the phantom obstacles appearing in the robot's world model in the vicinity of the roll-up door.

The obvious test of this theory was to place the robot in a fixed position about 6 feet from the door, and then raise the door and see if the problem went away. The robot moved into position, the door was raised, and to the utter amazement of all save perhaps the robot, all five sensors responded with bad data! Worse yet, the data were consistently bad rather than just erratic.

This was one of those rare moments in R&D where adrenalin pumps madly, logic is defied, and it is painfully apparent that we as humans are not as smart as we like to think we are. Something obviously was amiss, yet what could

it be? The robot had been operational with ultrasonic ranging capability for 7 years. It had operated daily in this very building for the past 3, with never even the slightest sign of a problem of this type. Yet there it was, sitting in the open garage door, looking out into the warm San Diego night at an empty parking lot, with all five of its ultrasonic ranging sensors seeing ghost targets at distances of 2 to 11 inches. The night was calm, with not a breath of air stirring, and deathly quiet except for the chirping of the crickets....

The crickets??? There obviously were no residual echoes when pinging into a zone totally devoid of targets. The other previously mentioned possibility for acoustical interference was some external source of ultrasonic noise. Such an occurrence had never been observed in years of operational evaluation of ROBART, however, except when deliberately attempted with the discharge of a pressurized air hose approximately two feet from the sensor array. Why was external interference suddenly becoming a problem now?

Perhaps the answer lies in the nocturnal nature of the cricket, which hides during the day, and comes out looking for food at night. While ROBART had on occasion operated all night in an unattended fashion, it had been within the confines of the lab area, which was environmentally controlled and tightly sealed to keep out the corrosive salt air from the nearby Pacific Ocean. Odds were there were no crickets in the lab area, but what about the rest of the building? The robot was moved back to the center of the bay, the garage door closed, and the lights were then turned off. Sure enough, the sound of a lone cricket within the confines of the building shortly ensued, to be followed in turn by the chirping of at least two more. During this serenade the robot, which is capable of no-light operation as a mobile security detection and assessment system, was unable to move for fear of running into perceived obstacles in all directions! An intruder that stopped by the local bait shop first could disable a quarter-million dollar security robot with a bucket of crickets!

Before commencing some in-depth research into spectral analysis of cricket emissions, and documenting this startling discovery in a torrent of writings, it was deemed prudent to validate the assumption with one last test. On the following day, the robot was driven to precisely the same location in the bay, facing the closed garage door. If the problem did not appear during daylight hours when the crickets were silent, then the case would appear fairly strong. As the door was raised, however, exactly the same erroneous results became immediately apparent, yet not a sound could be heard other than the sequential firing of the ultrasonic transducers in the array. The cricket theory had fallen through.

Attention now turned to the garage door itself, as the sonar readings were valid prior to raising the door. Exactly what effect had this had on the system? It was decided to look directly at the range value calculated by CPU #3, the sonar controller, while moving a cooperative target away from the sensor in a controlled fashion. Surprisingly, the results at this level in the hierarchy were valid at all ranges from 1 foot to the maximum range of 35 feet. A glance at the CRT displaying range values at the Planner, however, showed a consistent error as soon as range exceeded 34 feet: the displayed output began to increase from zero, accurately reflecting measured range minus 34 feet.

This error, obviously, was a roll-over bug in the software. A quick check of the code running on the Scheduler showed the problem. The sonar computer (CPU #3) calculates ranges in double precision in increments of a tenth of an inch; the scaling routine used to trim the sonar data from two bytes to a single byte for transmission over the RF link to the Planner had a bug in it. The routine simply shifted the data in both bytes to the right by 4 bits and threw away the upper byte, effectively dividing by 16, for range increments of 1.6 inches. As long as the original range value was less than or equal to $0fff (34.1 feet), everything worked fine; a

range reading of $1000, however, would scale to $00, or zero.

Interestingly enough, the particular subroutine involved was dated January 1987. The bug had been buried for 3 years of extensive testing, simply because the robot had never been operated in an area which allowed the ranging system to see out to a distance greater than 34 feet. The open bay area had been specifically cleared to make room for the dead reckoning test described above, and for the first time, the error cropped up, significantly hampering the ability of the robot to perform. The layout of the bay provided more open space near the garage door (figure 17), leading to the incorrect assumption that the door itself was somehow contributing to the problem. Finally, as the number of people working in the space decreased towards the end of the day, more open space was subsequently made available within the same physical area in which the system had been performing earlier without incident, further masking the nature of the problem.

This preceding scenario illustrates an important point in the development of mobile systems, in that as the number of interactive microprocessors goes up, along with the complexity of the associated software, the potential for errors rises accordingly. Any change in the robot's surrounding environment is likely to introduce new conditions which can cause hidden problems to surface. The most significant changes ever to be encountered will occur when a system is moved from the laboratory out into the real world. In the foregoing example, engineers were slow to suspect a software error, because the ultrasonic ranging system had been operational for 7 years. Even the routine which contained the bug had been extensively employed for 3 years without incident. Accordingly, it is important to exercise the robot in strange and unfamiliar territory in order to flush out problems of this sort, which can then be methodically isolated and corrected through a systematic diagnostic approach.

3.3.3 Short Range Proximity Sensing

A converging type (Everett, 1988b) near-infrared proximity sensor is installed on the front and each corner of the base of the unit (figure 8), positioned to detect the presence of the floor, to preclude falls down stairways or other attempts to traverse unsuitable discontinuities which could entrap or even damage the unit. Any undesirable conditions detected by these sensors will cause the drive to be immediately disabled, and the Scheduler is alerted to which corner sensor detected the problem. The Scheduler then asks for help from the Planner, and re-enables the drive if a high confidence solution can be found, else speech synthesis is used to request human assistance (section 3.5).

Three Banner Multi-beam CX1-6 medium-range near-infrared proximity detectors (Banner Engineering Corporation, 1989) are arranged in a forward-looking horizontal array for collision avoidance purposes. Two additional units (Mini-beam model SM312D) are mounted on the left and right sides of the front panel of the lower base unit. These Banner modulated-beam sensors have adjustable maximum ranges, set for this application to about 30 inches for the CX1-6, and 15 inches for the SM312D. The proximity sensors provide extended protection capability in the direction of travel, and collectively can discern if an obstruction is directly ahead, to the right, or left of centerline. These sensors serve as a complementary backup to the collision avoidance sonar array, as discussed later in section 4.1.

An additional 35 dual-zone near-infrared proximity detectors are employed on the robot for collision avoidance. Pulsed output from two high-power light emitting diodes (LEDs) operating at 880 nanometers is reflected from nearby objects, and sensed by a photodiode detector (figure 18). An optical bandpass filter in conjunction with a differentiating network in the input circuitry effectively discriminates between

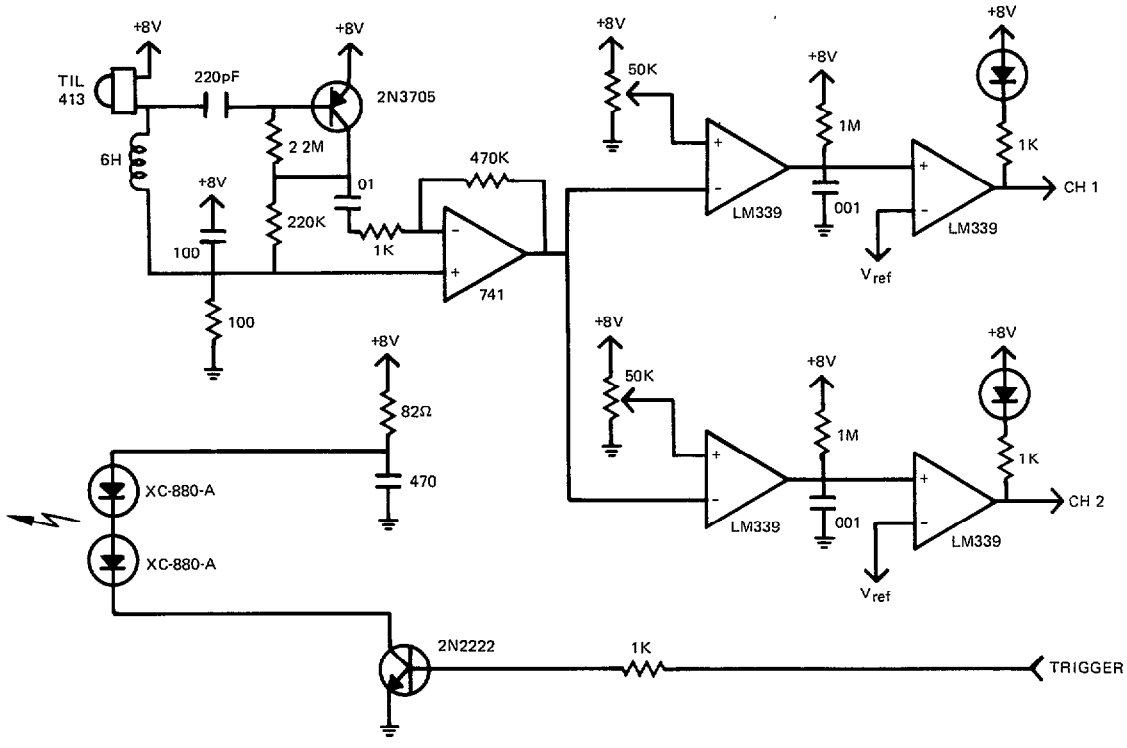


Figure 18. Schematic diagram of the dual-range near-infrared proximity sensors employed for collision detection.

valid returns and ambient noise. Two individually adjustable threshold detectors indicate when objects move to within 36 or 18 inches.

The proximity sensors are arranged in vertical columns on the body trunk and base of the robot, and the outputs within a given column logically θRed together to reduce processing overhead. This places a premium on angular as opposed to vertical resolution. (The robot is little concerned with the height of an obstruction, but rather where the obstruction is in the X-Y plane.) Sensors are placed to yield a higher coverage density in the forward direction of travel, while still providing a full 360-degree protected envelope.

Advantages of this type of sensor over ultrasonic ranging for close-proximity object detection are numerous. There is no appreciable time lag since optical energy propagates at the speed of light, whereas up to a full second can be required to update a sequentially fired ultrasonic array of only 12 sensors. In addition, optical energy can be easily focused to eliminate adjacent sensor interaction, thereby allowing the sensors to be fired simultaneously. Finally, the shorter wavelengths involved greatly reduce problems due to specular reflection, resulting in more effective detection of off-normal surfaces (Everett, 1985).

3.3.4 Long Range Proximity Sensing

A special programmable near-infrared proximity sensor was developed specifically for use on the prototype robot (Everett and Flynn, 1986), to gather high resolution geometric information for use in navigational routines. The primary purpose of this head-mounted sensor is to provide precise angular location of prominent vertical edges, such as door openings. A Polaroid ultrasonic ranging sensor is used in conjunction with the system to provide range data (figure 19).

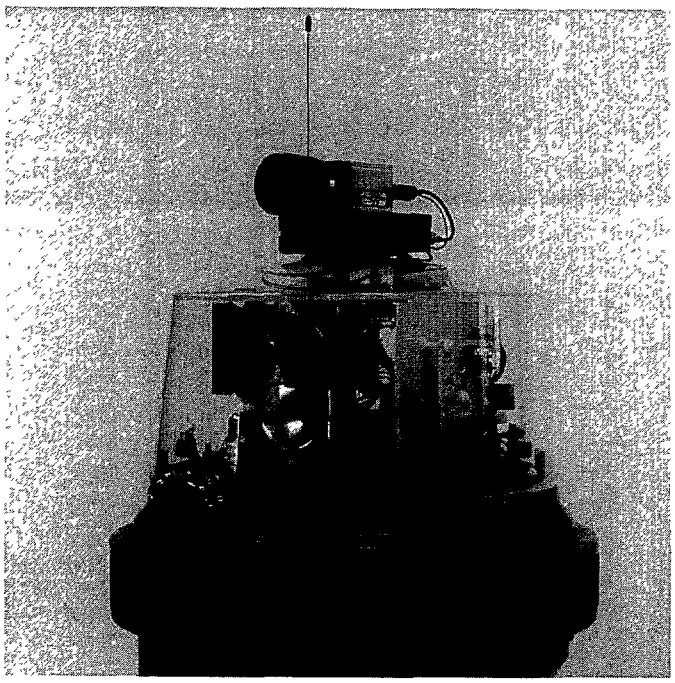


Figure 19. The parabolic reflector is used to focus returning energy on the detector of a programmable proximity sensor used to obtain high angular resolution data for navigation. Just below is the complementary Polaroid sonar ranging transducer.

An astable multivibrator produces a square-wave train of short-duration pulses, driving high-power XC-880-A gallium-aluminum arsenide LEDs, which emit energy in the near-infrared spectrum. The system uses an array of adjacent LEDs for increased range and sensitivity, with reflected energy focused on the lens of a TIL413 photodiode by a parabolic reflector. The output of this photodiode is passed through a L/C differentiator network, amplified, and fed to four separate follow-on threshold detector stages (figure 20). The receiver sensitivity is broken into four discrete levels by these individually adjustable threshold comparators. A strong return will cause all four channels to go low, whereas a weak return will cause only the most sensitive channel to indicate detection. No range information is made available, other than what can be inferred from the strength of the returned energy.

Unfortunately, the varying reflectivities of different surfaces preclude return signal strength from being a reliable indicator of distance. This turns out to be more a function of surface topography than of surface color; varying surface characteristics create uncertainties that thwart attempts to establish a practical correlation between signal strength and target distance.

Effective range is controlled by firing combinations of LEDs; thereby emitting regulated amounts of energy (i.e., the more LEDs illuminating the scene, the farther the detection range). The number of LEDs in the array that are enabled at any given time is specified by a microprocessor (figure 21), providing programmable control over the amount of transmitted energy, which in turn fixes the maximum range of the sensor. (The total number of active emitters can be any value between 1 and 4.) The robot *feels* around out to a distance of 5 or

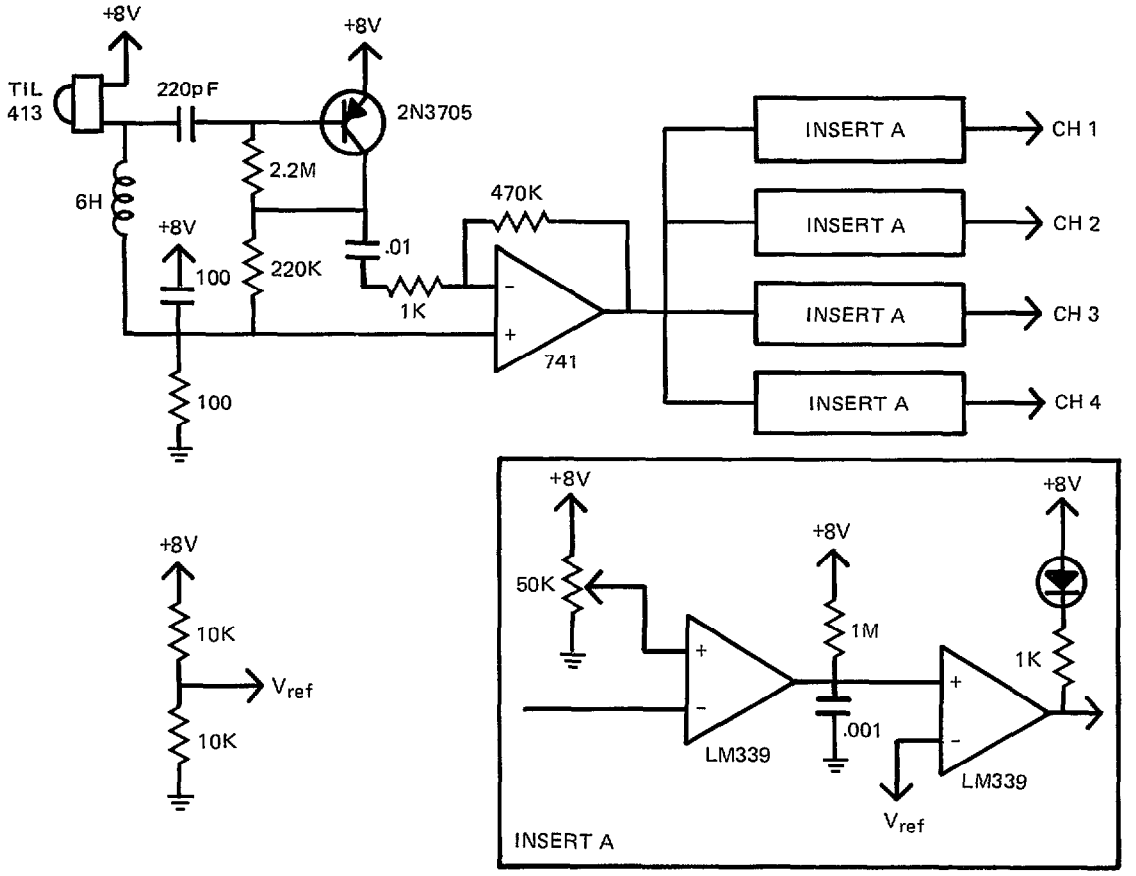


Figure 20. Schematic diagram of the receiver portion of the near-infrared proximity sensor.

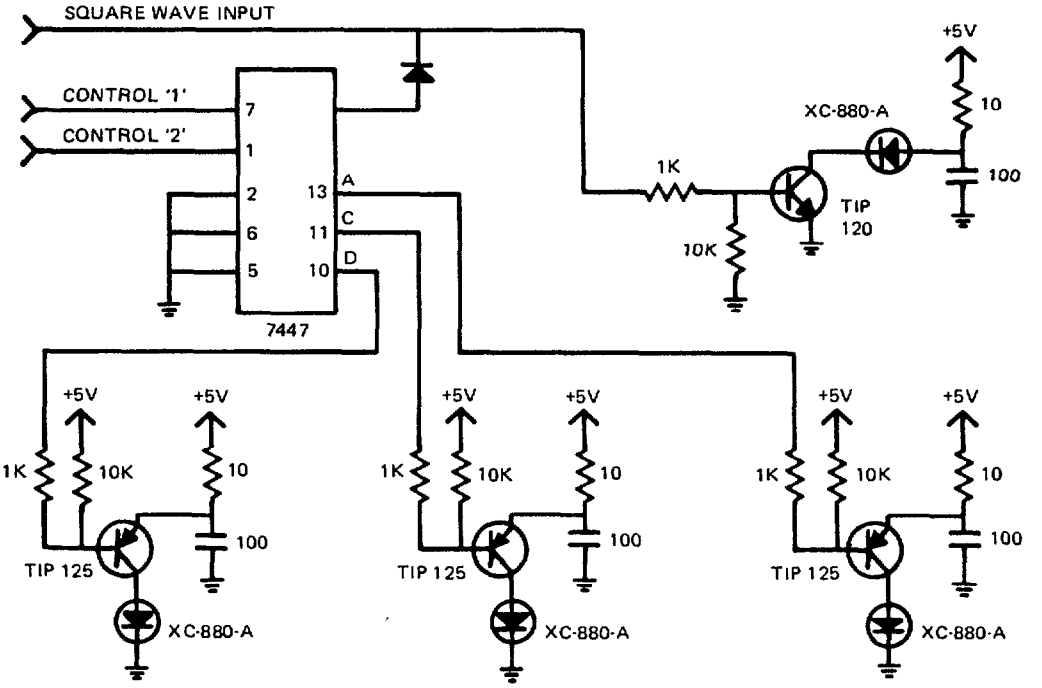


Figure 21. Schematic diagram of the near-infrared LED array controller.

6 feet, and notes those regions that are obstructed. The range of the sensor is extended a few more feet, and those areas showing no reflected energy are probed again. This process is repeated at computer speed until the sensor has mapped the entire region in terms of range discontinuities as a function of bearing, out to its maximum possible range.

During experimental testing the system proved capable of seeing out to an average of 6 feet with one LED active, 10 feet with two LEDs active, 13 feet with three, and a maximum average range of 15 feet attainable with all four.

3.3.5 Stereo Vision

A specially configured stereoscopic vision system (figure 22) initially provided for additional high resolution data acquisition, and was the robot's alternate means of locating and tracking a homing beacon on the recharging station (section 3.5). The system did not represent a true three-dimensional capability, however, in that each of the cameras consisted of a horizontally-oriented linear (as opposed to two-dimensional) CCD array.

The twin cameras in effect provided no vertical resolution, but furnished range and bearing information on interest points detected in the horizontal plane coincident with their respective optical axes, 44 inches above the floor. This limitation was consistent, however, with the two-dimensional simplified world model employed by the robot, wherein objects are represented by their projection onto the X-Y plane, and height information is not taken into account.

A structured light source was employed in conjunction with these stereo cameras for ranging purposes. A 6-volt incandescent lamp was pulsed at about a 10-Hz rate, appropriately lensed to project a sharply defined V-shaped pattern across the intersection of the camera plane with the target surface. This feature effectively eliminated the classical stereo correspondence problem, and improved system performance when viewing scenes with limited contrast. The incandescent source was chosen over an active laser diode emitter because of simplicity, the response characteristics of the CCD arrays, and the limited range requirements for an indoor system.

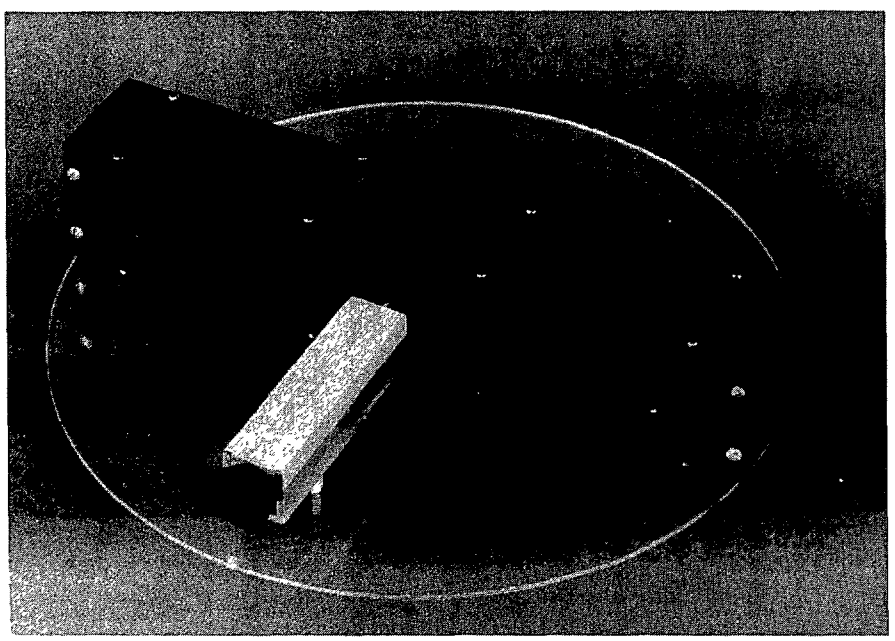


Figure 22. An active stereoscopic vision system employing two linear CCD array detectors was originally employed on ROBART II to obtain high resolution geometric data for navigation and intrusion detection.

In addition to providing geometric information for navigational purposes, the vision system served as both a primary and secondary indicator of a security violation. In the primary mode, the scene under surveillance was digitized, and the resulting template stored for reference. At normal system frame rates (60 Hz), subsequent scenes were captured and compared pixel by pixel to this template; any deviations outside of a specified tolerance signified a change indicative of intrusion. By periodically resetting the reference template at specified appropriate intervals, natural and gradual changes in scene illumination went unnoticed, whereas sudden deviations were immediately detected.

In the secondary mode, the cameras were trained on an area of suspected disturbance, and the video images analyzed for signs of motion between frames. The centroid of any region thus identified was calculated, providing a highly accurate bearing in camera coordinates to the intruder's location. A transform to robot coordinates was then performed based on the current head position, and the head would turn to center the image. This system was removed in FY 88 to accommodate a more versatile two-dimensional CCD camera operating in conjunction with a video line digitizer, to be discussed in section 3.4.2. This combination allows for live video to be relayed back to the console operator for use during teleoperation and security assessment, yet preserves the ability to do line-oriented image processing locally onboard the robot.

3.3.6 Wide Angle Optical Ranging System

This section describes a relatively simple and inexpensive near-infrared ranging system being developed for NOSC by Quantic Industries, Inc., San Carlos, CA under the Small Business Innovative Research (SBIR) Program (Moser, Everett, 1989). The system is not intended for installation on ROBART II, but will be employed instead as a collision avoidance ranging module on the follow-on third-generation robot depicted earlier in figure 3.

The prototype unit was designed around the following general guidelines:

Range acquisition over a field-of-regard of 100-degrees azimuth and 30-degrees elevation

Realtime range measurements out to 7 meters at up to a 10-Hz update rate

Small size and weight, low cost, and minimal power consumption

Rugged and maintainable, with no moving parts

Active triangulation ranging is employed with about 5-degree spatial resolution over a nominal field-of-regard of 100-degrees azimuth and 30-degrees elevation. Under typical indoor conditions, fairly accurate target detection and range measurements are obtained to about 8 meters in the dark and about 5 meters under daylight conditions. No mechanical scanning is employed, and the entire field-of-regard can be scanned in 0.1 to 1 second, depending upon the required accuracy, allowing range measurements to be taken in realtime while the robot is in motion.

The transmitter consists of 164 high-power, gallium-aluminum-arsenide LEDs mounted in an array behind a spherical lens, so as to produce a corresponding number of narrow, evenly spaced beams that interrogate the field-of-regard. The LEDs in the array are sequentially activated at a particular repetition rate, and a synchronous receiver detects reflected energy from targets within its field-of-view (FOV). A portrayal of the structure and the projected light beams are shown in figure 23. The LEDs are self-lensed to yield relatively narrow beams so most of their power is projected within the critical angle of the sphere lens for high power transfer efficiency. The pattern of the beams and their positioning behind the lens for the desired 5-degree spatial sampling is presented in figure 24.

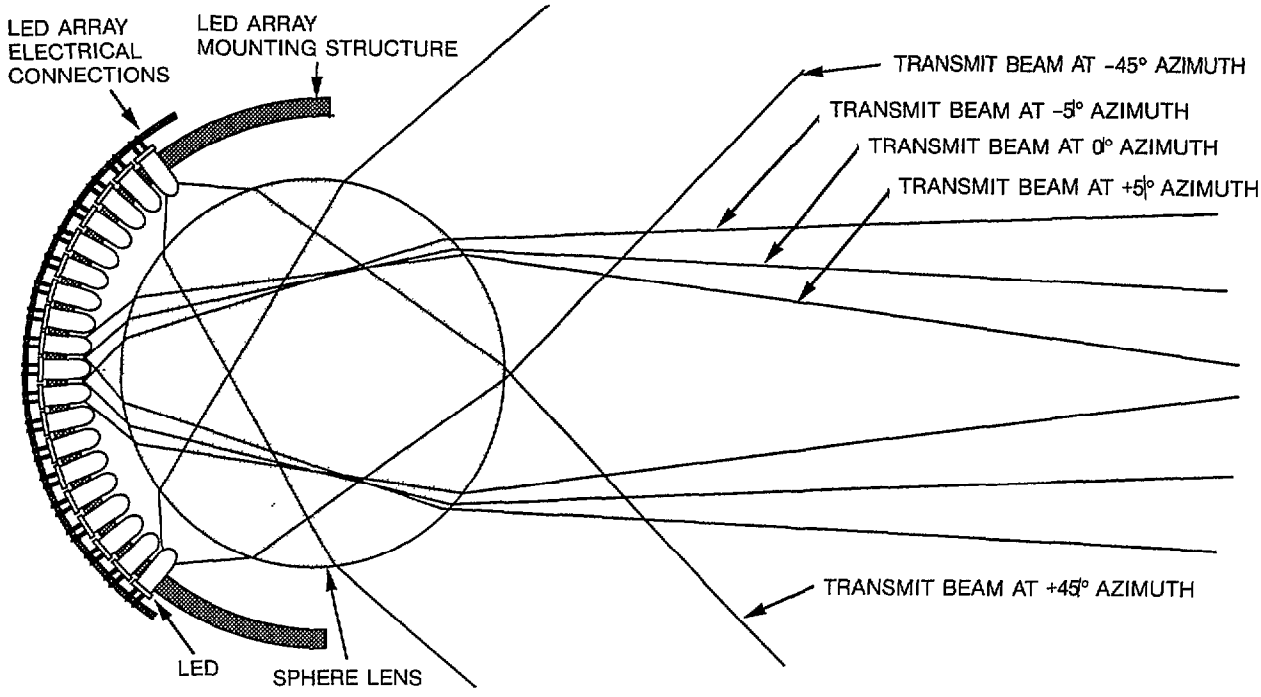


Figure 23. Physical configuration of the near-infrared LED array and spherical lens employed in the transmitter portion of the wide-angle optical ranging system developed by Quantic.

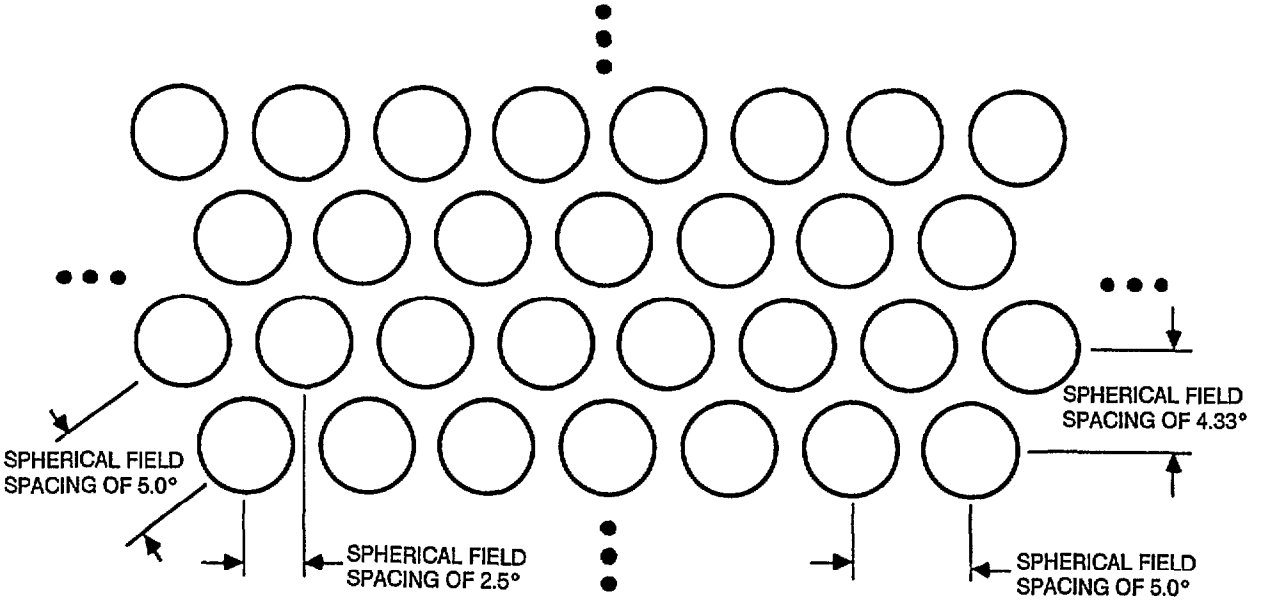


Figure 24. Partial diagram of LED array layout behind spherical lens of figure 23.

The optical receiver consists of two identical units, each covering a FOV of about 50 by 50 degrees. Both units contain a Fresnel lens, an optical bandpass filter, a position-sensitive detector, and the associated electronics to process and digitize the analog signals. The receiver uses a silicon lateral-effect position-sensing photodetector to measure the location (in the image plane) of transmitted light reflected (scattered) from a target surface. The transmitter and receiver are vertically separated by a 10-inch baseline.

The location of the centroid of reflected energy focused on the position-sensing detector is a function of the particular beam that is active and the range to the target being illuminated by that beam. The position signals from the detector (resulting from the sequential activation of LEDs in the transmitter) are collectively processed by a dedicated microcomputer to determine the ranges to valid targets throughout the sensor's FOV. Target azimuth and elevation are a function of the position of the LED (in the transmitter array) active at the time of detection. A look-up table derived from calibration data is used to perform the position-to-range conversions and to compensate for receiver nonuniformities.

3.4 SECURITY SENSOR SUBSYSTEMS

3.4.1 Off-the-Shelf Sensors

A simple form of passive detection capability intended primarily for indoor scenarios can be illustrated by the use of a microphone which allows the system to *listen* for sounds in the protected area. Figure 25 shows the circuitry currently employed on ROBART II; an automatic gain control (AGC) feature in the amplifier stage adjusts to ambient conditions, and any subsequent increase in signal level is detected by the LM 339 comparator. (This system will be discussed in more detail in section 3.4.3.) Vibration monitoring sensors are an extension of this concept, and are usually coupled to the floor through wheel contact when deployed on a mobile platform.

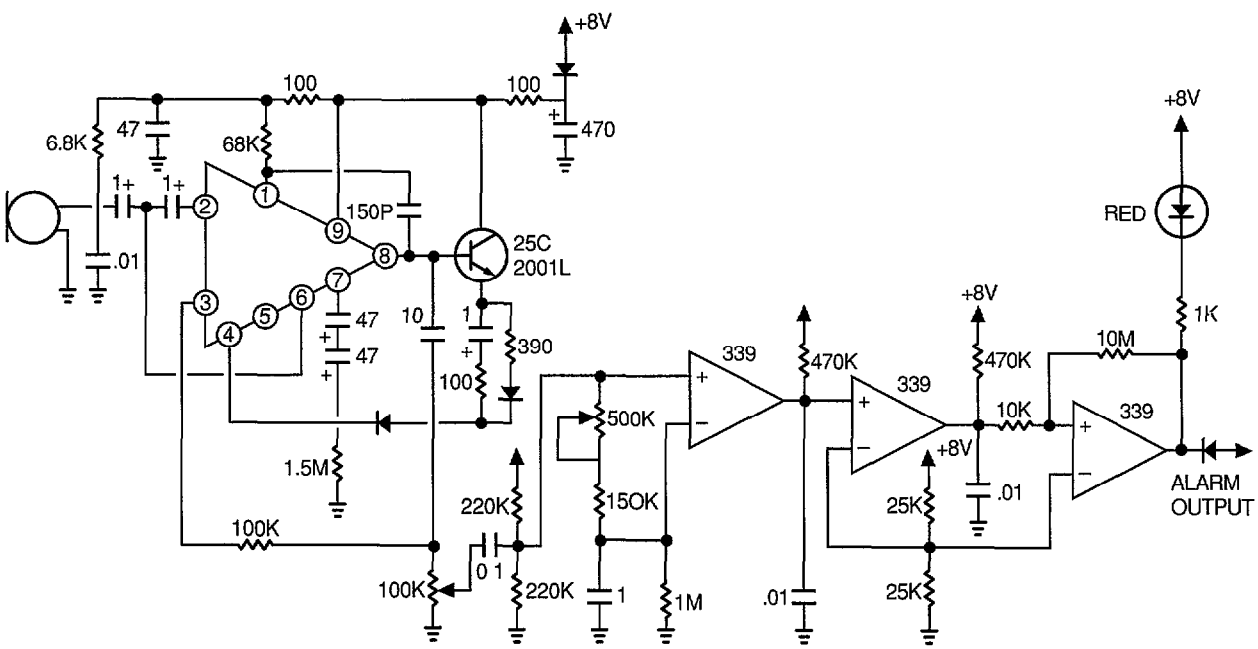


Figure 25. Schematic diagram of one of the three acoustical intrusion detection sensors employed on ROBART II.

An example of an optical motion detector which responds to changes in perceived light level is seen in the D-1072 integrated circuit manufactured by Sprague, which incorporates a built-in lens to create a cone-shaped detection field. After a brief settling period upon power-up, the circuit adjusts itself to ambient conditions, and any subsequent deviations from that setpoint will result in an alarm output. The low cost and directional nature of the device allow for several to be used collectively in an array to establish unique detection zones which help locate the relative position of the suspected security violation. The ability to provide geometric resolution of the intruder's position can be invaluable in tailoring an appropriate response in minimal time.

A significant development in detection technology is seen in the passive infrared motion detector. Originally designed for both indoor and outdoor fixed-installation security systems, this type of pyroelectric sensor quickly found application on mobile robots due to the small size, low power consumption, and excellent performance and reliability characteristics. The principle of operation as a motion detector is essentially equivalent to the optical sensor described previously, except a different wavelength (10 micrometer) in the energy spectrum is being sensed (Everett, 1988b). This type of detector exhibits a low nuisance alarm rate in indoor environments, but can be triggered by gusty wind conditions when employed outdoors.

Microwave motion detectors operating at radio frequencies rely on the Doppler shift introduced by a moving target to sense the relative motion of an intruder. The electromagnetic energy can penetrate hollow walls and doorways, allowing the sensor to *see* into adjoining rooms in certain circumstances. This feature can be used to advantage by a robot patrolling a hallway to check locked office spaces and storerooms without need for entry (Everett, 1988b).

Vision systems offer a more sophisticated method of sensing intrusion in outdoor as well as indoor applications, with the added benefits of excellent resolution in the precise angular location of the intruder. A surveillance camera can be used to digitize a scene for comparison with a previously stored image pattern representing the same region, and significant deviations between the two can be attributed to motion within the FOV. *Windowing* techniques can be employed on most systems to selectively designate certain portions of the image to be ignored (such as a tree blowing in the wind), resulting in a significant reduction in nuisance alarms. Simple algorithms that distinguish purposeful from random motion can further refine this discrimination feature at the intelligent sensor level (Everett, 1988b). Calculated boundaries of the perceived disturbance within the image frame can be used to automatically reposition the camera so as to be centered on the activity of interest.

3.4.2 Reconfigurable Video Line Digitizer

Virtually all applications involving image processing today are consistently bound to conventional practices which are heavily influenced by available hardware. The first step of course is to acquire the image. Traditional approaches employ a *frame grabber* which consists of the necessary circuitry to convert an entire two dimensional image into a corresponding digitized array which can then be stored in computer memory. For every pixel there must exist an associated location in memory which describes scene intensity (gray levels) at that particular location. For a conventional 525-line television image with 512-pixels-per-line horizontal resolution, this equates to 268,800 memory locations (bytes). In other words, over a quarter megabyte of memory is required to store a single frame of video.

The second step involves processing the data that has just been acquired. Due to the array size, even the simplest of operations, such as thresholding to convert to a binary image, is time consuming. More elaborate operations simply cannot be done using simplistic hardware before the next image arrives at the NTSC rate of 60 frames/second.

There are some applications, however, which do not require all of the data captured by a conventional frame grabber. A good example of such an application is video motion detection such as was introduced in section 3.3.5. Experiments with the linear CCD arrays showed in certain physical security scenarios, it was possible to detect motion by examining only one horizontal line which cuts through the region of interest in the scene. If several horizontal lines equally spaced throughout the scene could be easily acquired, effective full screen coverage could be achieved without the need to *grab* the entire frame. The image processing needs would be greatly reduced, and in most cases could be performed during the wait period between lines of interest when the acquisition system was idle. As an example, if only a single line is sufficient, the memory requirement would be reduced to half a kilobyte, with 16.4 milliseconds available for processing before the next line must be digitized.

The Reconfigurable Video Line Digitizer was developed for just this purpose, and consists of a high speed (100-nanosecond conversion time) analog-to-digital (A/D) converter which samples the composite video signal of a conventional NTSC format camera output. The composite video is also fed to a sync separator (figure 26), which splits off the horizontal and vertical sync pulses and provides a frame index. (The AD9502BM is a single-chip video digitizer available from Analog Devices (Hansford, 1987), which contains a flash A/D converter and integrated vertical and horizontal sync strippers). The horizontal sync pulses drive a counter which identifies the horizontal line of interest in the scene, whereupon line digitizing is performed. The digital output of the A/D converter is written directly into dual-buffered high-speed (35-nanosecond) video RAM, in order that it might be accessed later by the microprocessor when the A/D is idle.

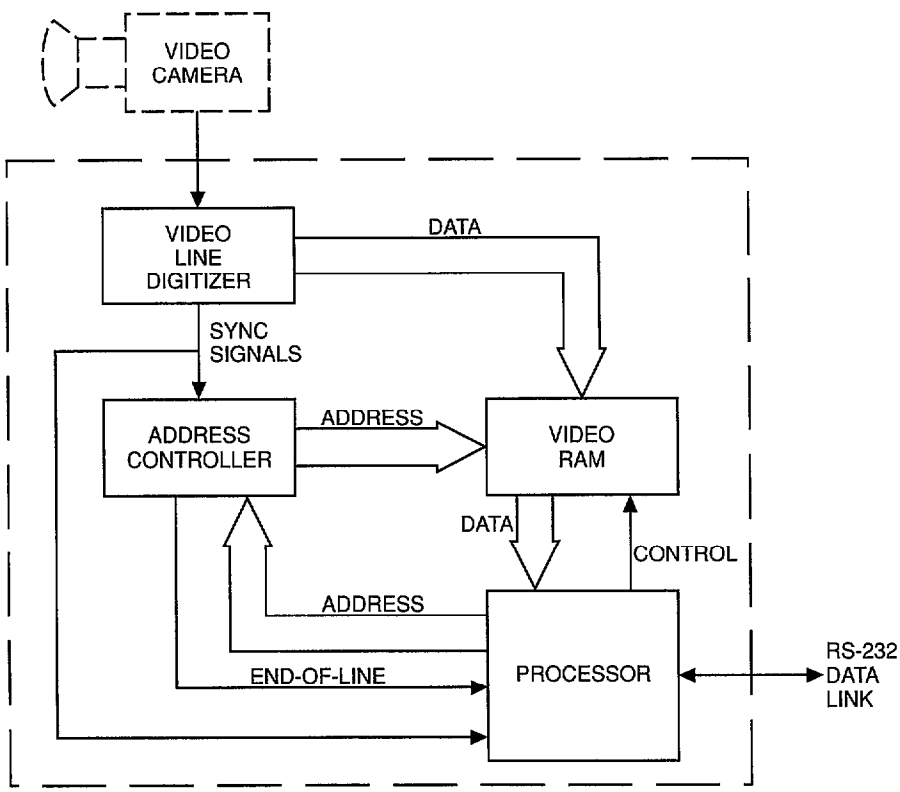


Figure 26. Block diagram of the Reconfigurable Video Line Digitizer developed for use on mobile systems where battery power and computational resources are limited.

The line digitizer consists of three basic subcomponents: (1) the video digitizing section, (2) the video RAM storage, and (3) the controlling microprocessor. Its basic function is to perform a flash A/D conversion of a composite video signal for a specific line of interest within a video frame, store the results in high-speed video RAM, and then transfer the results to normal storage for analysis before digitizing the same (or a different) line in the next frame.

Three counters are required in order to implement this capability. The first requirement is to select the desired line for digitization. This is accomplished through use of an external event counter accessed by the firmware resident on the controlling microprocessor. This counter is reset on each new video frame by the associated vertical sync pulse, and incremented by the arrival of a horizontal sync pulse at the beginning of each line. A second counter is needed to identify the start of actual video data at some predetermined time after the arrival of the horizontal sync pulse (figure 27), in order to ensure that digitization begins at the proper point. The third counter is required to identify the end of the data in order to properly terminate the digitization sequence at the end of the line. The latter two counters are implemented in hardware as opposed to software in order to achieve the necessary quick response.

As shown in the lower right corner of the block diagram of figure 26, the initial delay counter is driven by the pixel clock output of the AD9502 video digitizer, and reset by the horizontal sync pulse. The length of the desired delay is preset on the DIP switch. The pixel counter is then started after the initial delay counter times out, and begins to increment the address of the video RAM as the digitizer successively stores the results of the A/D conversions across the line. This process is repeated for each new line in the frame, with the previous values overwritten by the new data for the current line. When the external event counter reaches the specified line number, the process is halted with the desired gray scale values stored in video RAM. At this point the system switches

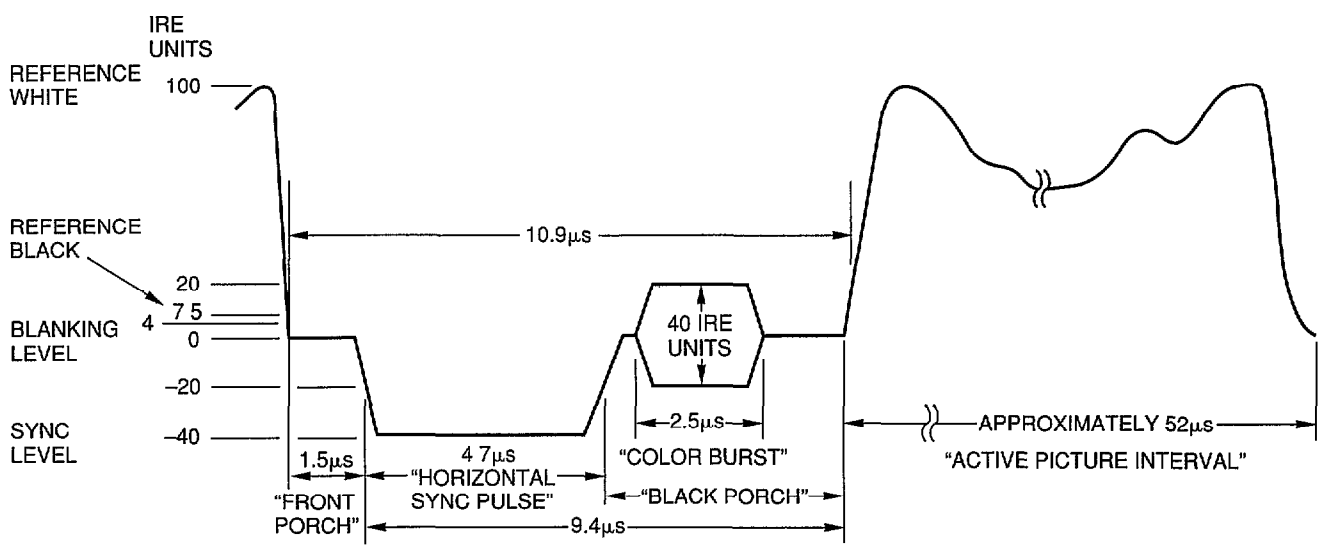


Figure 27. Portion of an NTSC composite video signal which illustrates the relationship between the horizontal sync pulse and the active picture interval (Hansford, 1987).

from the digitizing mode to the processing mode, and the microcomputer transfers the contents of the video RAM into normal system RAM for subsequent analysis.

The vision processor, CPU #6, commences data transfer and analysis when the *line grabber* has completed an acquisition operation. The most simplistic motion detection scheme involves subtracting the current intensity array from a previously acquired array, and reacting to any significant discrepancies between the two, which are indicative of a change in the scene under observation. In reality, some software filtering is required to eliminate noise and reduce the occurrence of nuisance alarms, but this is easily accomplished on a 512-element linear data array in the time available. (For simple motion detection schemes, 256 elements of horizontal resolution are more than adequate, further reducing required system complexity.)

Assuming full 512-pixel coverage, only 2K bytes of RAM are sufficient to support the microcomputer operating system and to save three select lines of video data, which normally would be equally spaced across the region of interest. Once motion is detected in any of the three lines, it is possible to select new lines for the next motion analysis operation. If these lines are chosen in such a fashion around the vicinity of the initially detected disturbance, it is possible over successive frames to converge on and effectively bound the area perturbed by the intrusion. In this fashion, the system can detect and output information describing the geometric area involved so as to provide servo-control inputs for camera positioning or robot motion algorithms.

3.4.3 Passive Acoustic Array

An acoustic detection array (ADA) intended to provide bearing information to the source of detected noise was developed in FY 89, based on the passive hearing sensor circuitry presented in figure 25 of section 3.4.1. The array consists of three omnidirectional microphones symmetrically oriented 120 degrees apart, and separated by a distance d. A block diagram is presented in figure 28. The ADA is mounted on top of

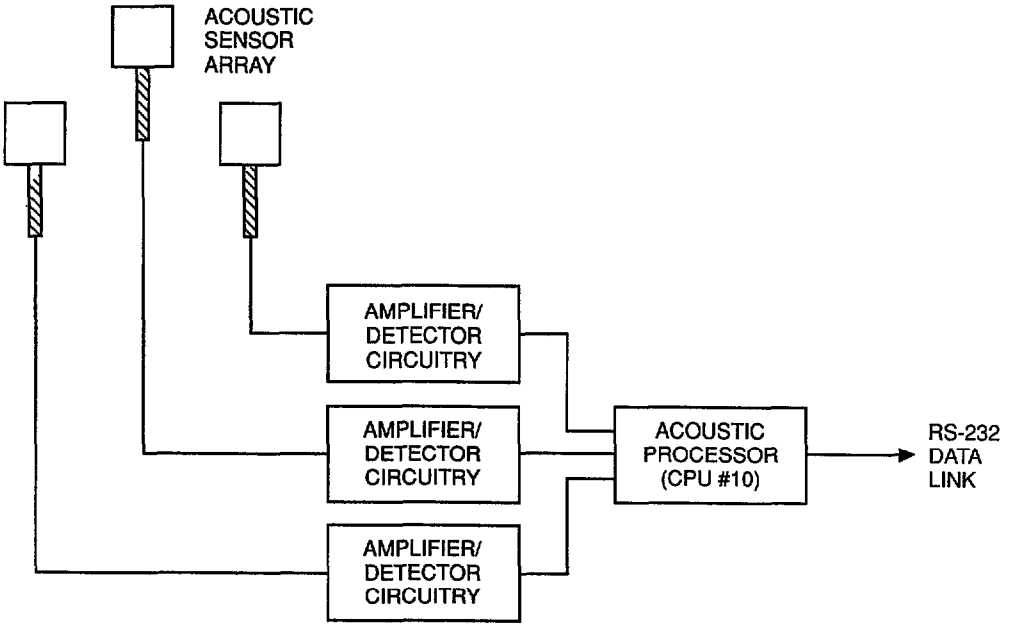


Figure 28. Block diagram of the Acoustic Detection Array which passively calculates a bearing to the source of a detected audible disturbance.

ROBART II, with the three transducers individually supported by coil springs. The springs provide some degree of acoustical isolation, while raising the transducers so as to yield a clear path for wavefront propagation without any blockage by the video camera (figure 29).

The ADA will calculate a bearing to an acoustical disturbance when the sound travels across the array and triggers all three detection elements in a specific sequence, the exact order of course being dependent on the relative position of the source. Because of the symmetrical orientation discussed above, we can classify the direction of the disturbance as being in one of six sectors by examining the *firing* sequence of the comparators associated with each of the three detectors.

Each sector is bounded by two lines, the first extending from the array center 0, through the first firing sensor (S_i), and a second originating at 0 and passing through a point OB_{xy} midway between the first two sensors detecting the incoming noise. The subscripts x and y are taken from the first and second sensors to trigger, as depicted in figure 30. Table 1 lists the firing sequence information for each sector.

Assuming the intruder (source of sound) is some distance away from the robot when initially detected, we can neglect the difference between the robot's height and that of the source with little adverse effect on resultant accuracy, and consider all derivations in a two-dimensional plane.

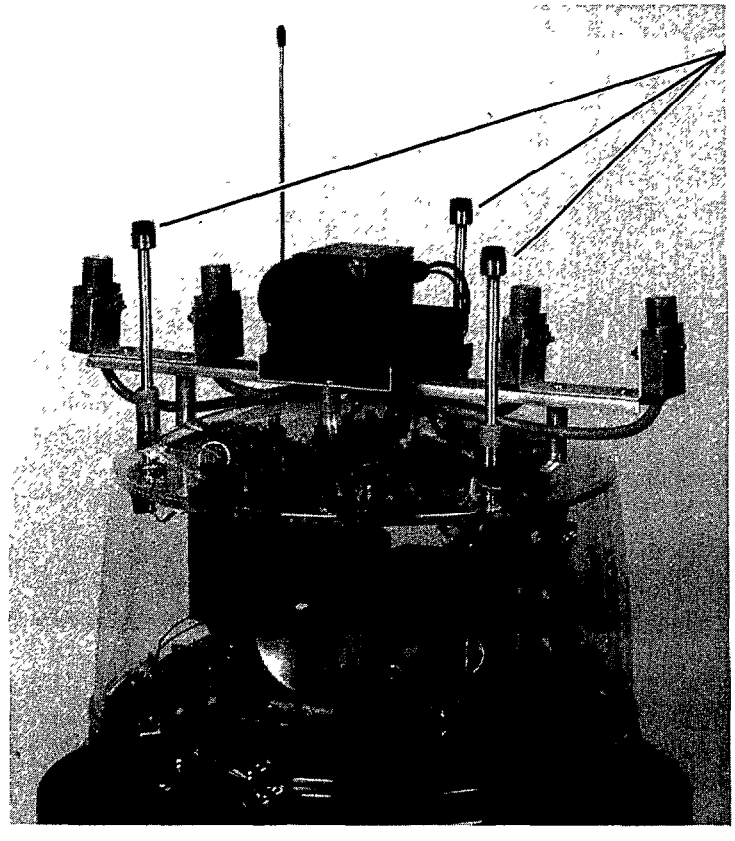


Figure 29. Three spring-mounted omnidirectional sensors are employed in the passive acoustical detection array mounted on the head as shown in this photo.

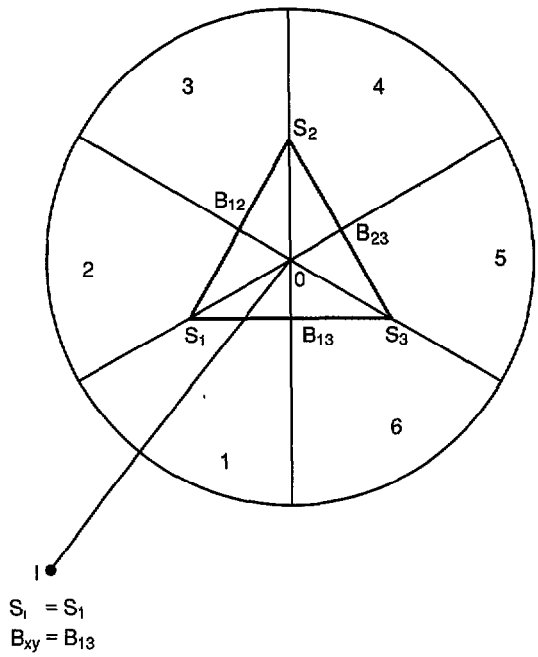


Figure 30. Diagram illustrating the relationship in the horizontal plane between acoustic sensors S1, S2, and S3 and the corresponding six sectors 1 through 6.

Table 1. Sensor firing sequence for the six potential sectors.

Sector #	Firing Sequence Information 1st firing	2nd firing
1	sensor #1	sensor #3
2	sensor #1	sensor #2
3	sensor #2	sensor #1
4	sensor #2	sensor #3
5	sensor #3	sensor #2
6	sensor #3	sensor #1

The individual sensor outputs are active low, with a negative transition triggered by the arrival of an incoming noise. Referring to figure 31, delay time T_1 (between 1st and 2nd firings) and delay time T_2 (between 2nd and 3rd firings) can be measured in order to determine the angle to the source, as discussed below. In keeping with the previously stated assumption that the intruder is some distance away from the sensors when first detected, wavefront propagation can be simplistically modeled as parallel lines perpendicular to a line extending from the array center 0 to the source of the detected disturbance. In addition, the speed of sound in air is assumed to be constant over the region of travel involved.

For each sector, therefore, it is possible to calculate a bearing to the location of the sound source, relative to the line segments OB_{xy} and to OS_i, the line segment of the corresponding sector, where i is the sector index. Figure 32 shows the relationship of angles θ_i and ϕ_i in the first sector, for sound propagation traveling from a source located at point I towards the array center 0. From this it can be seen that

$$\theta_1 = \angle B_{13}0I \tag{13}$$

$$\phi_1 = \angle IOS_1 = \angle S_3S_2C \tag{14}$$

and from triangles S_3S_1A and S_3S_2C:

$$\sin\theta_1 = \frac{T_1V}{d} \tag{15}$$

and that

$$\sin\phi_1 = \frac{T_2V}{d} \tag{16}$$

therefore

$$T_1 = \frac{d\ \sin\ \theta_1}{V} \tag{17}$$

$$T_2 = \frac{d\ \sin\ \phi_1}{V} \tag{18}$$

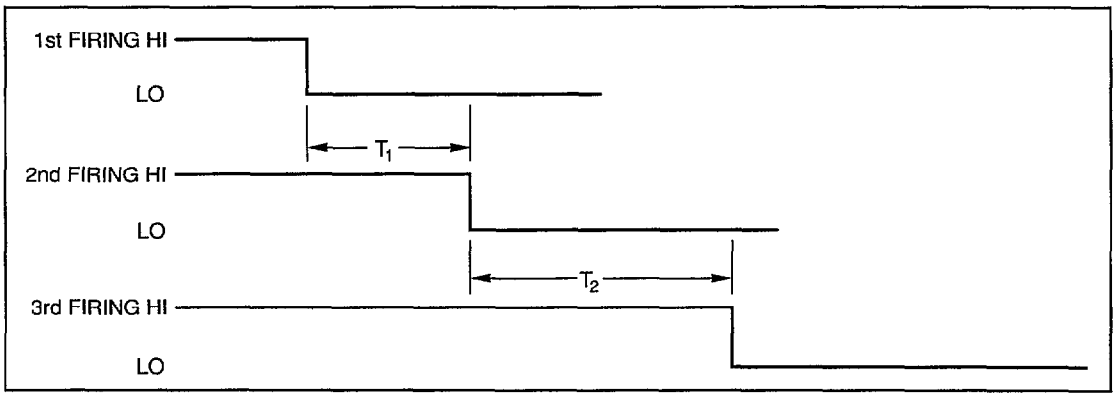


Figure 31. Timing relationship between delay times T_1 and T_2 is used to calculate bearing to source.

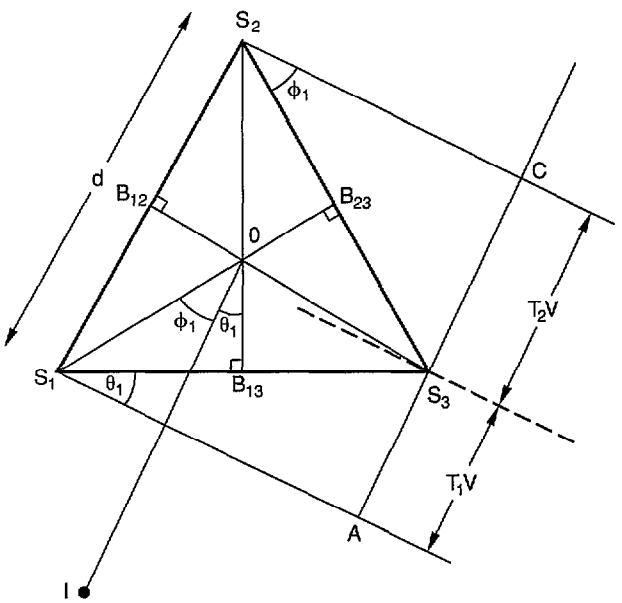


Figure 32. Relationship of angles ϕ_1 and θ_1 in the first quadrant for sound propagation from source at I towards array center O.

where

V = speed of sound
d = distance between array elements

and

$$\theta_1 + \phi_1 = 60°$$

which yields

$$\frac{T_2}{T_1} = \frac{\sin \phi_1}{\sin \theta_1} \tag{19}$$

$$= \frac{\sin (60° - \theta_1)}{\sin \theta_1} \tag{20}$$

and since

$$\sin(a - b) = \sin(a)\cos(b) - \sin(b)\cos(a) \tag{21}$$

equation (20) becomes

$$\frac{T_2}{T_1} = \frac{\sin 60° \cos\theta_1 - \sin\theta_1 \cos 60°}{\sin\theta_1} \tag{22}$$

and since

$$\cos^2(a) + \sin^2(a) = 1 \tag{23}$$

we have

$$\frac{T_2}{T_1} = \frac{0.866\sqrt{1 - \sin^2\theta_1} - 0.5\sin\theta_1}{\sin\theta_1} \tag{24}$$

which reduces as follows

$$\frac{T_2}{T_1} = 0.866 \sqrt{\frac{1}{\sin^2\theta_1} - 1} - 0.5 \qquad (25)$$

$$\frac{T_2}{T_1} + 0.5 = 0.866 \sqrt{\frac{1}{\sin^2\theta_1} - 1} \qquad (26)$$

$$\left(\frac{T_2}{T_1} + 0.5\right)^2 = \frac{0.75}{\sin^2\theta_1} - 0.75 \qquad (27)$$

Therefore

$$\sin\theta_1 = \sqrt{\frac{0.75}{\left(\frac{T_2}{T_1} + 0.5\right)^2 + 0.75}} \qquad (28)$$

Similarly, θ_i can be calculated in the other sectors, using the appropriate values of T_1 and T_2.

To avoid using floating point math, we scale the ratio T_2/T_1 (by shifting to the left 4 bits) and treat it as an integer, which still provides enough resolution for an accuracy of ±3 degrees. T_2/T_1 serves as the index for a 225-element lookup table containing the appropriate values (0 to 60 degrees) for θ_i. With this calculated value of θ_i, and knowing the specific sector involved (from table 1), a relative bearing to the source can now be easily computed as shown in table 2. Note this bearing is reported relative to the robot's head position (pan) angle. Figure 33 shows angles θ_i in their corresponding sectors.

Table 2. Derivation of bearing to source by sector number.

Sector	Bearing to Source
1	$+\ \theta_1$
2	$+(120° - \theta_2)$
3	$+(120° + \theta_3)$
4	$-(120° + \theta_4)$
5	$-(120° - \theta_5)$
6	$-\ \theta_6$

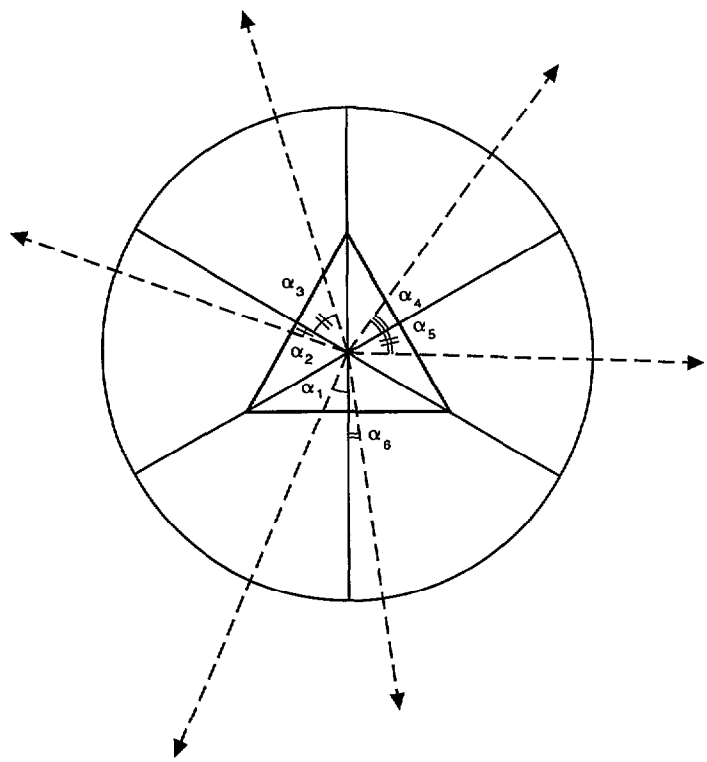


Figure 33. Angles from potential sources to array center O for each of the possible six sectors.

3.4.4 Dual Element Optical Motion Detector

The Sprague D-1072 optical motion detector presented in section 3.4.1 suffered from two significant drawbacks which limited the utility and contributed to its eventual discontinuation: (1) the current consumption of the device was too large to make battery-powered operation an achievable reality, and (2) it responded only to visible light. The fact that the chip was incapable of sensing in the near-infrared region of the optical spectrum means that an intruder using an active-source night vision device would not trigger an alarm even if he pointed the high-power source directly at the sensor at point blank range. There are no systems in place even today at high-security facilities employing elaborate automated security equipment which warn guards that the area is being illuminated by near-infrared energy.

Therefore, in support of the security application addressed in this research effort, a dual-element optical motion detector was designed specifically for scenarios in which the guarded installation could be under observation by potential intruders armed with a night vision device employing an invisible near-infrared source. The output of a cadmium-sulfide photosensor, sensitive only to visible light, is compared to an integrated lagging reference voltage derived from the same sensor, such that any change in scene intensity above a specified threshold will be detected. An identical circuit monitors the output of a silicon photosensor shielded by a near-infrared optical filter.

Examination of the output states of both circuits is thus indicative of the type of lighting involved when motion is sensed (i.e., near-infrared, fluorescent, or incandescent). Fluorescent and incandescent light both produce energy in the visible-light portion of the energy spectrum, and will activate the cadmium-sulfide detector, which is not sensitive to near-infrared. Incandescent and near-infrared sources will penetrate the optical filter to activate the broadband silicon detector, but the fluorescent source will be blocked. The resulting truth table 3 applies:

Table 3. Sensed energy derived from detector status.

Activated Detector	Sensed Energy
Cadmium Sulfide	Fluorescent
Silicon	Near-Infrared
Both	Incandescent

3.5 AUTOMATIC RECHARGING

The automatic recharging function can be broken down into two specific subtasks: (1) locating and homing in on the recharging station, and, (2) making the physical connection. Numerous proposed solutions have been developed over the years; most of these, however, employ a favored and sometimes specific direction of approach, which significantly complicates the task of alignment and mating, and thus reduces the overall effectiveness.

One of the standard approaches to this problem calls for the alignment of a special plug on the robot with a mating receptacle on the recharger. This can be done, but requires complicated hardware as well as software, and thus is susceptible to many problems which collectively serve to reduce the system reliability. If the plug is designed to mate with a conventional AC distribution system, electrical shock is introduced as a potential hazard.

Another type of existing system consists of a rectangular platform with a semicircular cavity in one side, and a track tape lying on the floor, extending from the cavity. When a low battery condition is detected, the robot starts searching for the track tape by random motion aided by preprogrammed information about the operating environment. If the tape is found, it is followed to the recharging station, where connection is made with two contacts, one on either side of the cavity. This approach has several disadvantages. The robot wastes a lot of rapidly diminishing power searching for the tape. The tape itself must be physically installed on the floor, where it is subject to abuse. The recharging

station cannot be easily relocated without moving the tape and reprogramming the robot with information on where to search (unless purely random searching is employed). The approach path is constrained to a single direction as defined by the tape; objects which inadvertently block this path can impede a connection. Another problem can arise due to the nature of the mating contacts; if the robot is not perfectly aligned when entering the cavity, a good connection may not result.

Other approaches involve the use of coded near-infrared beacons for homing. One embodiment involves placing the beacon on the wall behind the recharging unit, which is situated on the floor below. In this particular case, errors can arise from any subsequent displacement of the recharging unit with respect to the beacon. Such displacement may result from human intervention (the charger is moved by a cleaning crew, etc.) or as a consequence of robot impact during the docking procedure. A second embodiment solves this problem by placing the beacon on an enclosure roughly the size of a telephone booth, which the robot then enters in order to effect a connection. Spring contacts mate with two circular bands around the robot's base to furnish the recharging current. This solution is obviously costly, less flexible in terms of relocation within the operating area, and more obtrusive. In either case, approach must be made within a narrowly defined window of access, which is a major drawback should this path become obstructed.

In the previous examples, the robot often has difficulty measuring the distance to its recharging station during the approach, due to problems associated with specular reflection of ultrasonic energy from smooth planar surfaces. An additional ambiguity is associated with the reflecting target, which could be a wall, and its actual relationship with respect to the charger.

Another problem common to all systems of this type is contact degradation (pitting) at the mating surfaces due to make-and-break action of the conductors, especially where heavy recharge current is employed. This situation is even further aggravated due to the mechanical bounce which occurs as a consequence of the impact, which can cause damaging power transients as well in the computer logic and control circuitry onboard the robot.

Figure 34 shows the physical structure of the recharging system used by ROBART II. At the bottom of the platform is a 1/8-inch thick 18-inch diameter aluminum plate mounted atop a 1-inch thick particle board base of the same diameter. Situated above this, and electrically

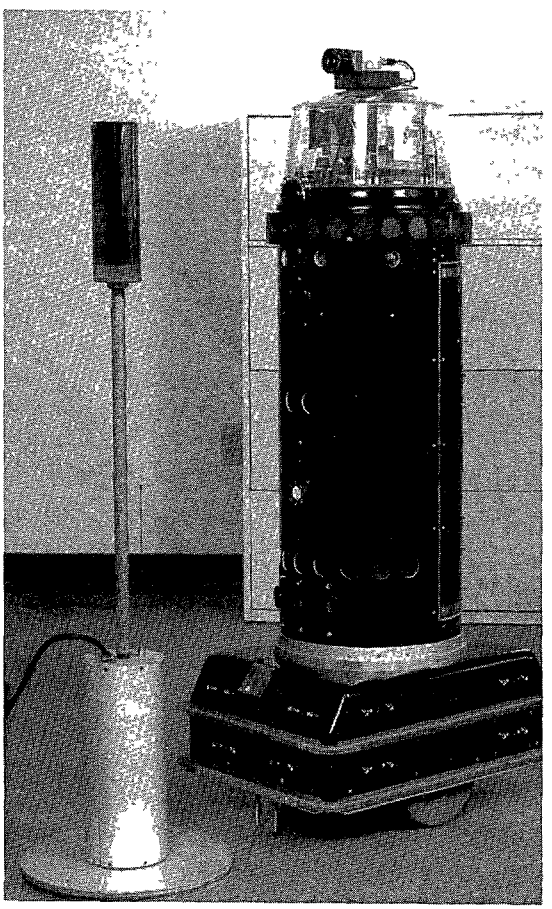


Figure 34. ROBART II homes in on the near-infrared beacon until spring-loaded contacts in the base and front bumper make contact with the baseplate and metal cylinder of the recharging station.

insulated by means of a Plexiglas spacer, is a 12-inch high cylindrical housing which contains the recharging and system power supplies. At the top of the unit is the homing beacon, visible

from any direction, and supported by a metal pipe with a coaxial extension which can be adjusted to match the height of the sensor on the robot. The homing beacon can consist either of a plurality of high-output near-infrared LED emitters equally spaced around the perimeter of a support ring, or a single vertical-filament incandescent lamp surrounded by an optical filter which blocks the visible portion of the emitted energy (Kodak Wratten 89B or equivalent), or a combination of the two. Enclosed within the cylinder is a small box which houses a radio receiver and associated decoding and control electronics. The homing beacon is activated by the robot upon command via the radio link.

There are two power supplies associated with the recharging station itself (figure 35). A relatively low-current source remains energized at all times, and supplies power to the radio receiver and associated electronics. This supply also energizes the recharger contacts through a current limiting resistor to a peak potential of 20 volts, which acts as the *sensing* voltage for the robot, allowing the Scheduler to know when the robot has made connection with the recharger. As soon as the battery has been connected as a load, this voltage will drop to around 14 volts. The voltage drop is sensed by a special window comparator circuit on the recharging station, which in turn activates the high-power battery charging supply after a 2-second delay. (If the voltage falls below the window comparator's lower limit, the power supply cannot be activated, providing automatic short circuit protection.) This second power supply furnishes the current required to recharge the battery, and is automatically shut off when the robot disconnects and the load is no longer sensed. The delay is incorporated to allow the mating contacts to debounce before power is applied, markedly reducing contact erosion and pitting due to arcing and burning. The longer time constant associated with power supply voltage rise (as opposed to connecting to already energized contacts) eliminates harmful voltage transients which could damage the robot's computers.

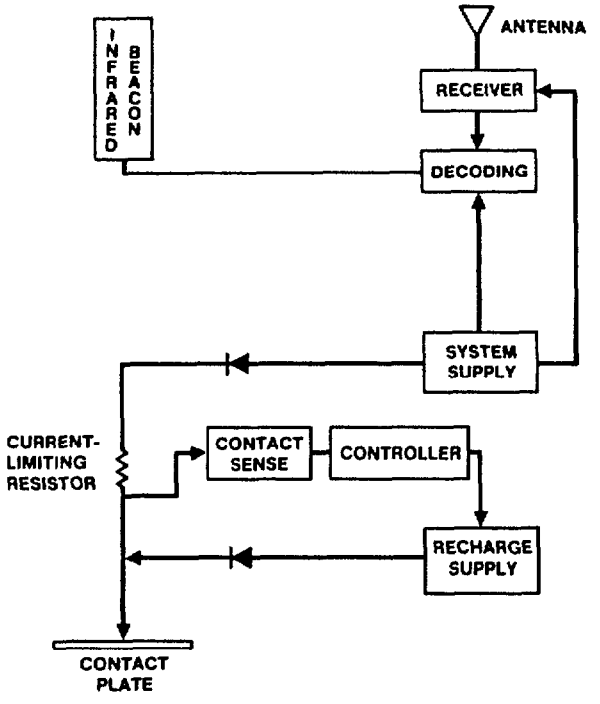


Figure 35. Block diagram of the automatic recharging system.

Once the battery monitor circuit on the robot detects a low-battery condition, the Scheduler initiates the recharging sequence by requesting a path to the recharging station. The path planner first draws an imaginary circle of 24-inch radius around the last known location of the charger, which is encoded in the model. A path is found to the nearest point of intersection with this line, and the robot moves to this position. The head pans from the full left to full right position while the room light intensity for each of 256 discrete head position angles is digitized and stored. At this point, the Scheduler activates a special near-infrared homing beacon on the recharging station via a dedicated radio link, and enters the beacon acquisition mode. The robot rescans the room from right to left, looking for a light source which was not present prior to beacon activation (positive identification can be achieved by turning off the beacon via the radio link while observing the selected target). Once the beacon is located, the Planner enters the new beacon position into the model if different from before, and the robot finds its

way to the station by an optical tracking system that keeps the head-mounted sensors centered on the beacon. The difference in output from two silicon photodetectors is used to servo the head position, and the resulting relative bearing used to calculate a steering command to cause the robot to turn in that direction.

Two sets of connections must be made to complete the recharge circuit, and will be referred to as HOT and GND. The task of connecting with the recharger is simplified by making these contact surfaces symmetrical with respect to the vertical pole which supports the homing beacon, so they present the same target to the mating contacts on the advancing robot, regardless of the direction of approach. The cylindrical metal housing at the base of the beacon support pipe serves as the point of contact for the GND leg, with the respective mating surface being a metal strip attached to the front bumper of the robot. The bumper is spring : loaded to absorb impact, and make possible the deflection of numerous microswitches used as tactile sensors for collision detection (section 3.3.1). This spring action serves to keep the bumper in tight contact with the housing once the two come together, compressing the springs.

The connection for the HOT leg is made through the mating of two spring probes with the circular aluminum base plate. The spring probes are vertically oriented extending downward from the edge of the robot chassis. As the front of the chassis passes over the plate, moving toward the metal housing supporting the beacon, the spring probes are brought into contact with the plate, and contact is maintained as motion continues toward bumper impact. The geometry of the configuration ensures the probes will be in contact with the plate as long as the front bumper contact strip is touching the power supply housing; therefore, considerable margin for alignment error is allowed, since the strip is 10 inches wide. As soon as bumper contact is made, the circuit is complete, and recharge current flows to the battery. A blocking diode on the robot ensures the probes are not at battery potential when the robot is not connected to the charger.

At this point, the robot motion is halted by the Scheduler, which senses bumper deflection via the microswitches, in conjunction with a signal indicating recharge power is sensed onboard. The contact sense circuit of the recharging station will activate the recharging supply after a 2-second delay. The robot software monitors the probes as well as battery level while the system is recharging, and should electrical contact be lost, will effect the necessary forward motion to re-establish the connection. The robot can deactivate the beacon via the radio link once initial contact has been made and the recharging process initiated, without affecting the recharger supply.

3.6 RF COMMUNICATIONS LINK

The current RF communications link consists of a single-board MMC-102 microprocessor manufactured by R.J. Brachman, and a REPCO Utility Data System RF modem, configured as shown in figure 36. The MMC-102 has one

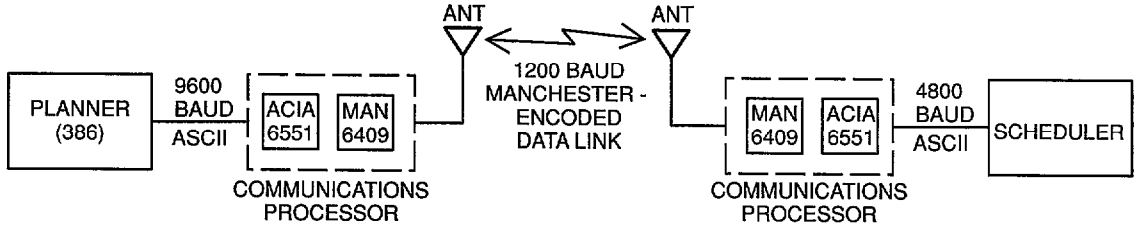


Figure 36. Block diagram of the current RF data communications system employed on ROBART II.

6522 VIA (IO2) removed to accommodate in its place a 6551 ACIA (Asynchronous Communications Interface Adapter); another 6522 VIA (IO1) provides an interface with an HD6409 CMOS Manchester Encoder/Decoder.

The RF link is configured to look like an ordinary RS-232 modem. The incoming ASCII message is circularly buffered in a 256-byte buffer until a complete packet has been received (table 4), and then retransmitted in Manchester format through the RF modem (table 5). A valid packet is considered to be all data between the exclamation point (!), start of packet, and carriage return (CR), end of packet. The RF modem is normally held in the receive mode, and switched to transmit mode when there is a complete data string ready for transmission.

Table 4. Packet format between the local ACIA and planner (or between remote ACIA and scheduler onboard robot).

Byte Number	Value	Meaning
0	!	Start of Packet
1	Ascii Hex	Inquiry value
2		
"	"	Data
"	"	"
"	"	"
n-1	"	"
n	CR	End of Packet

Table 5. Packet format between the Repco RF modems.

Byte Number	Value	Meaning
0	!	Start of Packet
1	Ascii Hex	CRC hi byte
2	"	CRC hi byte
3	"	CRC lo byte
4	"	CRC lo byte
5	Binary Hex	Packet number
6	Ascii Hex	Data
"	"	"
"	"	"
n-1	"	"
n	CR	End of Packet

The RF Link software is summarized in the flow chart depicted in figure 37. The current Repco modem is half duplex, and a significant delay is required when switching between transmit and receive modes. Experience has shown this can create some bottlenecks when high-speed interactive communications between the robot and host are required. This problem will be solved in a future link upgrade which will incorporate a higher speed (minimum 4800 baud) full-duplex RF modem.

Features common to the wireless communication systems considered include a standard RS-232 input/output interface between the host computer and MODEM, at full duplex data rates of 4800 and 9600 baud. For the current frequencies authorized, 408.1 and 417.7 MHz, frequency modulation (FM) is required for transmission; premodulating the data (channel coding) however, can create a data stream which when transmitted via FM demonstrates improved noise immunity. Further considerations include size, weight, and power consumption of the radio link residing in the mobile, robotic system.

One possible candidate is the CAT-4880, manufactured by CATRON, Inc. (Sharpesville, Pennsylvania), which offers a 4800-baud full-duplex FM wireless communication system, using either synchronous or asynchronous data transmission. As advertised, the CAT-4880 operates within the 450-470 MHz commercial frequency band; a system radiating in the military band of interest however, can be manufactured upon request. Both the base station and mobile unit include a stub antenna and antenna duplexer. The base station, designed for use in conjunction with the host computer, requires a 115-volt AC input and weighs 45 pounds. The mobile unit requires 12-volt DC input, weighs 5 pounds, and measures 6.5 by 11.5 by 2.25 inches. Transceiver characteristics include errorless reception for signal voltages in excess of 1 microvolt, with a rated transmit output power of

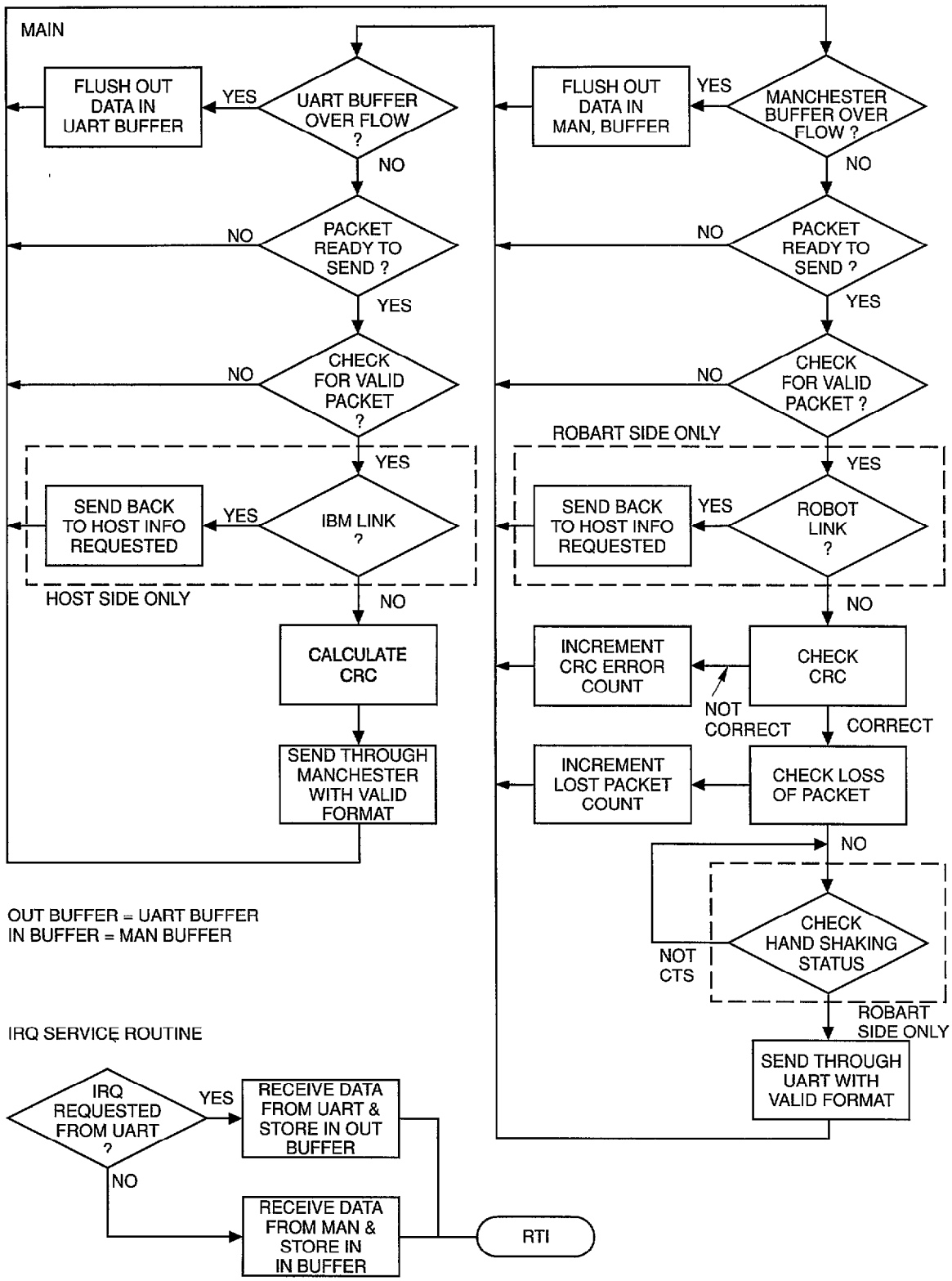


Figure 37. Flowchart illustrating the software structure employed on the communications processors (figure 36).

2 watts at 650 mA (150 mA in standby). The modulation scheme employed is straight FM, with no error correction/detection scheme at the RF side, nor is there any embedded channel coding algorithm. A polling or addressing scheme can be implemented in user software for multiple mobile unit operation. Cost of the base station is $3450, and $2975 for the mobile unit, or $6425 per host/robot communication circuit.

Individual modems manufactured by REPCO (Orlando, Florida) offer a variety of modulation schemes and data rates. Two of three potential candidates are the Radio Data System (RDS) 9600 (9600/4800 baud) and RDS 4800 (4800 baud) for synchronous data transmission. Unless otherwise noted, the following discussion concerns both modems, which are capable of full-duplex operation on the RF side with a built-in antenna duplexer. REPCO has implemented an FM scheme using coherent differential phase shift keying (coherent DPSK), where differential implies, in this case, differential encoding and decoding (detection). When using phase shift keying (PSK) modulation techniques, one defines an n-member signal set which effectively divides 360 degrees of available phase into n predetermined increments; the n phase-modulated signals thus created differ by 360/n degrees. The RDS 9600 uses signal sets consisting of 16 members for 9600 baud operation and four members at 4800 baud. The RDS 4800 uses an eight-member signal set. Both units advertise two watts radiated power, with a bit error rate (BER) of 1 in 1 million for receive signal voltages of 0.707 microvolts. Although an AC power supply is standard, a 12-volt DC source can be implemented for an additional $1100 per unit. The dimensions are 12.0 by 11.87 by 3.88 inches. These modems employ no RF error correction/detection nor channel coding techniques. The cost is approximately $5324 for the RDS 4800 and $6757 for the RDS 9600, which includes the cost of a DC source. For a host/robot communication circuit this comes to $9548 for the RDS 4800 and $12,414 for the RDS 9600, less antennae.

The third REPCO modem of interest is the Synchronous Link Radio (SLR) 96, for either synchronous or asynchronous data communication. Features which distinguish it from the RDS units include a frequency-shift-keying (FSK) modulation scheme, with the capability of either point-to-point or point-to-multipoint operation using a high speed polling technique. No RF error correction/detection nor channel coding techniques have been incorporated. The SLR 96 comes in two models. The host model is available with either 120-volt AC or 12-volt DC power source. Its dimensions are 2 by 10 by 10 inches, including the internally installed antenna duplexer. The utility model (SLQ) features a 12-volt DC source with an externally mounted antenna duplexer, and measures 5 by 7.75 by 3 inches. The current drain is less than 1 ampere while transmitting and 300 mA in standby. Unit cost is approximately $3000 for the host and $2850 for the SLQ, or $5850 for a host/robot communication circuit, less antennae.

In choosing between the three REPCO modems discussed above, one is faced with a choice in modulation schemes. The RDS 9600/4800 and RDS 4800 both feature coherent DPSK. The number of members in a signal space is inversely proportional to the minimum allowable signal-to-noise ratio (SNR) for a given BER. Comparing FSK and coherent DPSK modulation techniques, FSK possesses a 3-dB advantage in receive SNR for a given bit-error probability. Looking at the complexity of transmitter/receiver configurations required for detection, implementation of an noncoherent FSK detection system is simpler than that required for coherent DPSK (Sklar, 1988, and Oetting 1979).

Electronic Systems Technology (EST) of Kennewick, Washington manufactures two modems which are widely used by industry: the ESTeem Model 85 for simplex (VHF) transmission, and Model 86 (UHF) for half-duplex. Both these modems can be configured as repeaters, thus extending the range of reliable asynchronous communication. The packet with bit-compression RF data transmission scheme implemented in the ESTeem modems provides 16-bit CRC error checking and automatic packet retransmission for error correction services.

Using a Manchester-encoded FM modulation scheme and two watts of output power, a BER of one in 100 million is possible for signal strengths in excess of one microvolt. ESTeem Model 86 is available in the 400 to 500-MHz frequency band and requires a 12-volt DC power supply. Current drain characteristics are 700 mA in transmit, 300 mA in standby. The price for a single unit, not including antenna, is approximately $2180, or $4360 per host/robot communication circuit.

Faster transmission rates (i.e., 19.2 kilobaud and up) can be achieved by using direct-sequence spread-spectrum (DS-SS) modulation. A true spread-spectrum modem must meet the following requirements. First, the transmitted signal energy must occupy a bandwidth larger than, and be independent of the information bit rate. Additionally, demodulation must include correlation of the received signal with the spreading code used to modulate the information bit stream. Typically, the bit rate for the spreading code or chip rate is greater than the information bit rate. DS-SS modulation is accomplished by performing modulo-2 addition, at the chip rate, between the bit stream and chip sequence. The spreading code is typically generated using a shift register configuration; by varying the shift register initial condition, many different spreading codes can be generated from a single hardware configuration. Thus simultaneous transmission of information in the same frequency band is possible using code division multiple access (CDMA). Therefore, the use of identical DS-SS modems, each set to produce a different spreading code, allows for the simultaneous operation of multiple data links (Ziemer and Peterson, 1985). Additional advantages in using spread-spectrum modulation are greater insensitivity to noise and a lack of operator licensing requirements by the FCC. FCC guidelines provide for three frequency bands (0.902 to 0.928, 2.400 to 2.483 and 5.725 to 5.850 GHz) with the constraint that output power must not exceed 1 watt. Currently, DS-SS wireless communication links for data transmission are available in point-to-point as well as multipoint LAN configurations (Stover, 1990).

The Local Area Wireless Network (LAWN) manufactured by O'Neill Communications, Inc. (OCI, of Raliegh, North Carolina) offers a LAN configuration using AX.25 amateur packet-radio link-layer protocol, employing an eight chip-per-bit spreading code. The maximum throughput for this system is 19.2 kilobaud, a high data rate which somewhat compensates for the fact that full-duplex operation is not available. The system comes with an internal antenna array for omnidirectional coverage. Using manufacturer supplied software, it is possible to communicate with any one of 22 remote units, using one of three possible configurations: computer node, a modem node, or a printer node. When configured as a computer node, messages and files maybe exchanged with another computer; or data maybe sent to printer LAWNs. The printer LAWN configuration receives data from a computer LAWN and sends the data to the printer's serial data port. Initial investigation of a direct connection using a computer LAWN at the host end and a printer LAWN at the remote end yielded satisfactory results over distances up to 100 feet in a congested environment.

Although the LAWN advertises four channels, only one channel can be designated in the software installation phase. Accessing a different channel involves reinstalling the software on all LAWNs in the network. Each LAWN covers a frequency band of 26 MHz (902-928 MHz) divided into four channels, each with its own carrier or center frequency. Establishing a communication path is accomplished using a carrier sense multiple access (CSMA) scheme, where polling is conducted every 34 seconds. Output power is 20 mW, and a range of 150 feet can be anticipated for indoor (obstructed environment) use. This range can be extended by using a single LAWN in a repeater configuration; up to two repeaters can be used per communication path. Using a 12-volt DC power source, the LAWN draws 250 mA in both transmit and standby modes. The unit dimensions are 7 by 4 by 2 inches, and it weighs 1 pound. Cost is $298 per end including an antenna, or $596 per host/robot communication circuit. The capabilities and low unit cost of this product combine to

make the LAWN a potentially good candidate for establishing a short range RF communication link.

Agilis Corporation (Mountain View, California) also markets a DS-SS modulation device for wireless data transmission using a user-selected network protocol. Unfortunately, the communications module is compatible only with Agilis computer systems (Davidovici, 1990).

NCR Corporation (Dayton, OH) has just announced a plug-in board called WaveLan, intended for use on IBM PC/ATs. According to the preliminary claims, any number of personal computers can be linked together under the system, with a maximum range of 1000 feet in a typical office environment. The advertised data rate is two megabits per second, but actual testing has not been conducted at NOSC. (Repeated calls to NCR for additional information have gone unanswered.) The cost of a single board is $1,390, which includes an omnidirectional antenna housed in a 3-inch square case. For even greater security, an optional Data Encryption Standard security feature is available for $90.

Telesystems SLW, Inc. (Ontario, Canada) manufactures a wireless network available in two architectures: a point-to-multipoint network (ARLAN 100), and a connectionless PC LAN architecture (ARLAN 400). Only the point-to-multipoint architecture is discussed below. Due to the power output constraints imposed by the FCC, indoor operating range is limited to approximately 500 feet, but can be extended by using the ARLAN 150 repeater. A maximum of eight repeaters can be installed along any one communication path, at a cost of $2100 per unit.

All ARLAN equipment operates in the 902 to 928 MHz band, centered at 915 MHz, and uses a half duplex radio protocol with a 230 kilobit per second RF channel. A 100 chip-per-bit spreading code is used to achieve full use of the available bandwidth and maximum immunity against interfering sources and multipath. User port rates up to 19.2 kilobits per second, full duplex, are available.

The transceiver, ARLAN 010, includes one of three antenna types: a 3-inch monopole (standard), an omnidirectional whip, or a directional high-gain Yagi, and costs $2000, plus $80 for a nonstandard antenna. For single port devices, the ARLAN 130 Single Port Network Unit provides a network access point, and can link to any other ARLAN port using a port addressing scheme. The unit includes a transceiver function and costs $2000.

For host computer support, the ARLAN 110 Control Unit is capable of communicating with up to eight (and multiples of eight) remote devices at a time. This unit requires an ARLAN 010 Transceiver ($2000) and costs $1600. The ARLAN 110 implements an adaptive polling protocol to allow remote units to efficiently share the available 230 kilobits per second bandwidth. In a point-to-point link between two ARLAN 130 units, a CSMA access protocol is used. In addition, the user selects one of eight spreading codes for the network, allowing multiple networks to exist using Code Division Multiplexing (CDM) techniques.

For use on mobile robotic systems, the ARLAN 140 is small (4.3 by 1.5 by 9.2 inches) and weighs only 2 pounds. Costing $2000, the ARLAN 140 includes the transceiver function and an onboard rechargeable 12-volt DC battery pack (which can be replaced by a DC power supply onboard the robot). Current drain for the ARLAN 140 is 600 mA while transmitting a packet, 40 mA in idle. Idle mode is automatically set when there has been no transmission/reception activity for approximately 2 seconds. The preferred configuration is an ARLAN 140 used in conjunction with the ARLAN 130, which transmits a poll/query packet every 25 milliseconds. Both units employ 16-bit CRC error checking and automatic packet retransmission for error correction services. A single robot/host communication circuit costs approximately $4000, assuming the standard monopole antenna is used. Large areas can be covered through a technique known as Telesystems

Microcellular Architecture (TMA) which provides a seamless handoff of virtual circuit connections as a remote ARLAN unit moves from the area covered by one transceiver or repeater unit into the domain of another transceiver or repeater.

In summary, spread-spectrum communication technology appears to be much more suited to short range robotic applications than conventional modulation techniques. For ease in comparison, the systems previously discussed are listed in table 6 below in ascending order of cost. The information summarized includes only manufacturer and model, modulation scheme, approximate effective data rate and cost for each host/robot circuit.

Clearly, spread-spectrum modulation offers increased data transmission rates at significant cost savings when compared to conventional wireless transmission schemes. Of the two spread-spectrum communication systems discussed, the LAWN by OCI is perhaps most suited for a low-cost, single host/robot scenario. PC Magazine, however, in a comparison test of the LAWN and ARLAN systems, reported some noticeable interference to the LAWN from a nearby RF source, a wireless burglar alarm motion detector (Maxwell, 1990), most likely due to the smaller (eight chip-per-bit) spreading code employed. Accordingly, for more complex applications in which multiple host/robot circuits are required, the ARLAN system may be more appropriate.

3.7 SELF DIAGNOSTICS

During the development of any complex robotic system, particular attention must be paid to all issues affecting reliability and maintainability. It would be quite possible otherwise to create a problem bigger than posed by the original application being addressed, in terms of the skill levels necessary to service and maintain the robotic *solution*. The same technology which now enables the many new and innovative applications of flexible automation must be tasked with providing realistic solutions to their upkeep, especially in the case of systems which must operate in harsh and unstructured environments typical of Navy scenarios.

For this reason, considerable emphasis was placed throughout the development of ROBART II on providing the necessary inputs to support embedded self-diagnostic procedures. Each microprocessor in the hierarchy is assigned certain responsibilities, and has in turn control over all the necessary interface circuits needed to read in appropriate information and effect the necessary responses. The required subsystems are activated when needed, and de-energized afterwards to save battery power. Diagnostic procedures include those which are embedded in the code to run during normal operation, as well as secondary routines which can be called up by an operator to specifically isolate a suspected fault. Either category can be essentially divided into three parts: (1) read certain inputs, (2) analyze according to a rule base, and (3) effect a response.

Table 6. Summary of potential off-the-shelf RF communication link products in order of increasing cost.

Manufacturer/Model	Modulation	Baud	System Price	
OCI/LAWN	DS-SS	19.2K	$ 596	w/ antenna
NCR/WaveLan	DS-SS	2.0M	2,780	w/ antenna
Telesystems/ARLAN 100	DS-SS	19.2K	4,000	w/ antenna
EST/ESTeem 86	FM	4.5K	4,360	w/o antenna
REPCO/SLR 96	FSK	9.6K	5,850	w/o antenna
CATRON/CAT-4800	FM	4.8K	6,425	w/ antenna
REPCO/RDS 4800	8-PSK	4.8K	9,548	w/o antenna
REPCO/RDS 9600	16-PSK	9.6K	12,414	w/o antenna
REPCO/RDS 9600	4-PAK	4.8K	12,414	w/o antenna

The first of these needs is met in several ways. Quite a bit of information is available to the respective controllers onboard the robot as a consequence of their normal operation. For example, the head pan axis controller has constant access to head position, direction of rotation, angular velocity, etc., and can use this information to verify correct operation of all circuitry in executing a motion command. Similarly, the drive controller has such information available for both the left and right drive motor control subsystems. This information can be used to provide realtime *error detection* capabilities at the local level.

Extending the concept beyond error detection into the realm of true diagnostic capability, however, involves the acquisition of additional information normally not needed for the assigned control function alone. Referring back to figure 7 in section 3.2, there could be numerous reasons why a command to move the robot forward did not result in actual motion. A number of connectors are employed in the system, any one of which could prohibit operation if not properly seated. The PWM controller card could possibly have been removed for service. Each motor has a 10-amp circuit breaker which interrupts power in the event of overload. A *Drive Safety Lockout Switch* on the base of the platform can be used to de-energize the drive circuitry during maintenance operations. The RF *Emergency Abort* system allows an observer to halt robot motion by depressing a button on a small hand-held transmitter if an unsafe condition arises. Near-infrared proximity sensors are configured to disable the drive controller upon detection of a significant floor discontinuity, such as a descending stairway. And finally, there is the possibility of a failed component in any of the above drive-related subsystems.

The circuitry employed must therefore make provision for communicating status to the CPU which is tasked with running the diagnostic routines. The simplest information to analyze is binary in nature: the checkpoint reports *pass* or *fail*. Digital status can be read directly over an I/O port; analog information must be first converted to a digital value or in some cases a binary pass/fail bit. The previous examples cited as potential causes of a drive system malfunction will next be examined with respect to a means of instrumentation for reporting their status.

Cable connections are easily treated by dedicating one line in the multiconductor run to connector integrity, and grounding the remote end of this line via the terminating attachment, usually a circuit board. A single line can verify integrity for all connectors in series in the cable run if there is no need to specifically call out where the break lies in the event of a fault. On ROBART II, most of the interconnecting cable runs are 14- and 16-pin DIP jumpers, and so it is relatively easy to assign a dedicated line on each for diagnostic purposes, (figure 38). A pull up resistor at the test point will cause the input to float high in the event of a dislocated connector at either end of the jumper. If both connectors are properly seated, the diagnostic input will read at logic *low*.

Circuit board integrity is easily addressed by dedicating one line of the edge-finger connector to diagnostic duty, and grounding the respective mating finger to the circuit board ground bus. A pull-up resistor is used at the test point, so that the line floats high unless the board is correctly seated, and pulled low when the board is installed. It is prudent to employ a blocking diode as shown in figure 39 to preclude positive voltages from being fed back through this test point in the event the board is partially or incorrectly seated, or the wrong board is accidentally inserted. The sense line should be at the opposite end of the edge-finger connector from the trace which provides the *ground* connection for the board, so both ends of the board must be firmly in place in the mating receptacle before the sense line is pulled low.

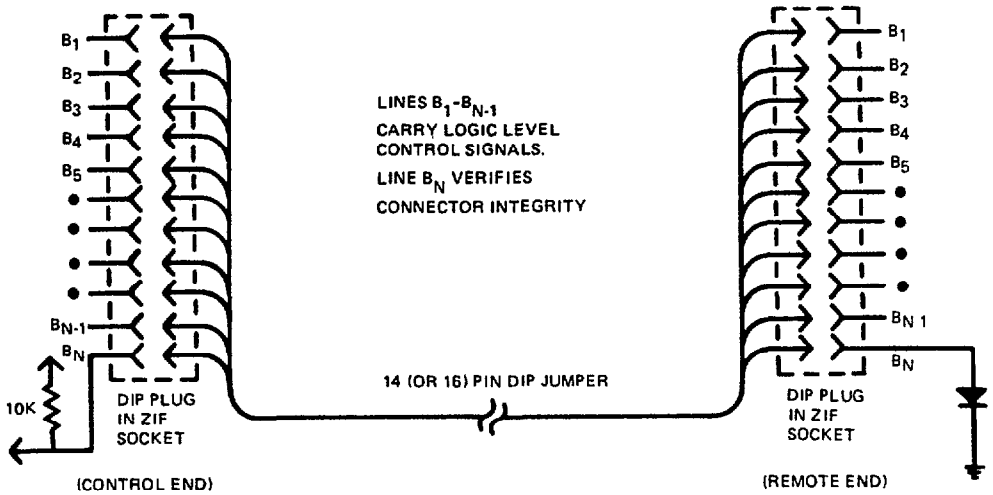


Figure 38. A single conductor in each of the 14- and 16-line ribbon cables used to interconnect various subsystems on the robot is dedicated to connector integrity diagnostic circuitry as shown here.

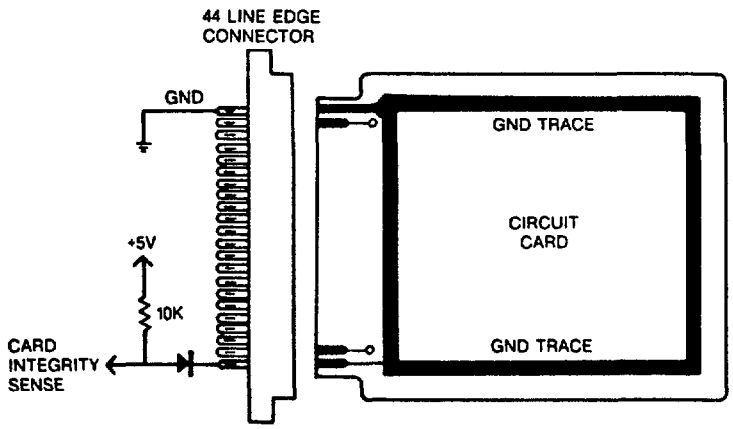


Figure 39. Proper circuit board seating is verified by checking one of the 44 edge connectors specifically dedicated on each card to the integrity sense function to ensure a logic low.

Switch position can be determined by using a double-pole switch for what normally would be a single-pole switching operation (triple-pole in place of double-pole, etc.) as shown in figure 40. The otherwise unused portion of the switch is wired to cause the sense line to change state from logic high to low when the switch position is changed by the operator (or actuating cam in the case of limit switches). Some standardization in design can be employed so the normal position of all switches is represented by the same logic level, which is a logic *low* in the case of ROBART. This facilitates writing of the diagnostic software, discussed later.

Breaker switches and fuses pose more of a problem, and a different monitoring technique must be employed. The actual voltage level must be measured on the load (output) side of the breaker (fuse), and compared to some previously established reference. This comparison is most easily accomplished with a voltage comparator, such as the LM 339, which is a quad package. The output of a voltage comparator changes state when the input voltage falls below the reference voltage. In this example, the reference voltage can be derived by scaling the voltage applied to the source (input) side of the breaker switch, (figure 41). This technique has

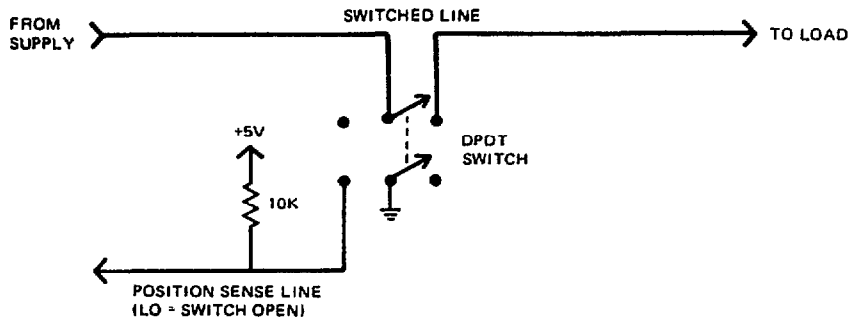


Figure 40. Switch position can be read by the diagnostic software through use of an additional piggy-backed switch function which toggles the logic level of the sense line.

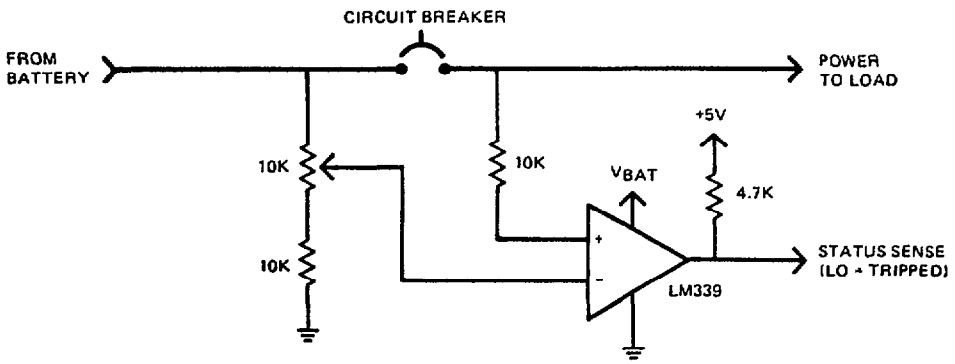


Figure 41. A tripped breaker switch can be identified by examining the output of a voltage comparator which compares the voltages on both sides of the device.

the advantage over a fixed reference in that momentary fluctuations as well as battery discharge over time will be automatically compensated. A newer type of breaker switch made by Airpax (Cambridge, MD) is now available which has a built-in mechanically-coupled SPDT status switch specifically designed for diagnostic purposes.

The status of special inhibit lines, such as employed in the RF *Emergency Abort* subsystem, can be read directly as long as they are logic level compatible (usually +5 and 0 volts). If not, such signals, provided they do not exceed 15 volts, can be easily converted to the appropriate level through use of 4049 (inverting) and 4050 (noninverting) buffers; the supply voltage provided to the buffer determines the output high level.

Missing pulse detectors can be employed to check for the presence of data streams, such as clock outputs, ASCII dumps, phase quadrature encoder outputs, the presence of PWM control signals, etc. Figure 42 shows the circuitry employed to sense drive motor operation by monitoring shaft encoder output. A similar circuit is used to verify output from the PWM circuit board.

Actual component failures are hard to pinpoint without going overboard on the instrumentation; it is generally preferable to isolate the fault to a certain area for more detailed troubleshooting by a technician. Nevertheless, certain critical components may warrant special

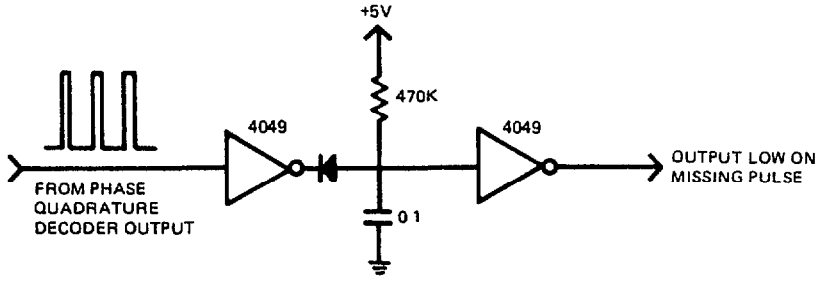


Figure 42. A missing pulse detector can be configured as shown to ensure the presence of a train of pulses in the output of a phase quadrature shaft encoder.

attention. Custom ICs such as the Dallas Semiconductor DS1232 can directly monitor the performance of CPUs, checking for clock rate, normal address- and data-bus activity, out-of-tolerance power supply conditions, etc. Voltage sources which supply interface circuitry can be monitored by window comparators set to flag an over-voltage or under-voltage condition (figure 43). Multiplexed analog-to-digital converters, such as the National Semiconductor 16-input ADC0816, can be configured with one input connected to a known reference voltage to periodically check operation.

Reading the various binary test points briefly described above is simplified from the standpoint of required I/O capability, component wiring, and associated software through the use of a special multiplexed input scheme based on the CMOS 4067 analog switch operated in the digital mode. As such, the 4067 essentially forms a 1-of-16 data selector: which of 16 inputs that is passed through to the chip output is determined by the binary word applied to the four input control lines. Figure 44 shows how the outputs of 16 of these devices are connected to yet another, in order to form a 1-of-256 data selector, which is driven by ROBART's auxiliary data bus. The lower nibble on the bus determines which input is activated for the low-level selectors 1 through 16. The upper nibble determines which of their respective outputs is passed along by the high-level selector 17 as a binary bit to the Scheduler.

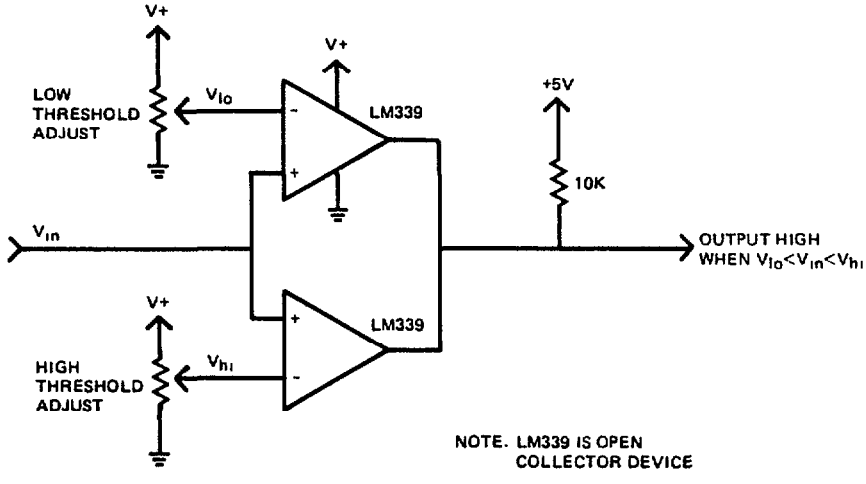


Figure 43. Window comparators can be employed to sense an out-of-tolerance power supply condition. The above circuit has separate threshold adjustments for over-voltage and under-voltage conditions.

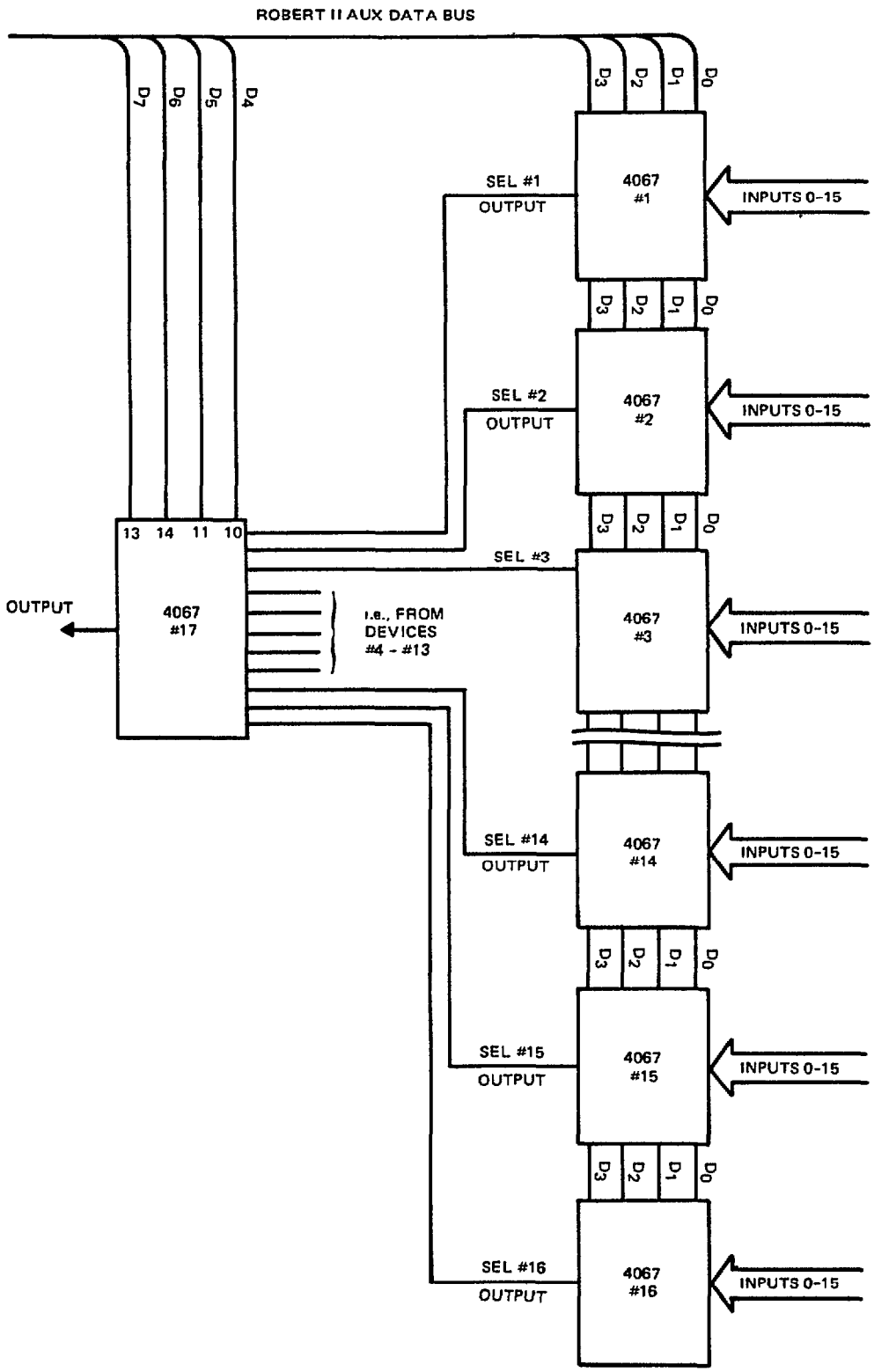


Figure 44. Seventeen 4067 analog switches are configured as shown to form a 1-of-256 input data selector to read the various binary diagnostic inputs on ROBART II.

Each of the first five low-level data selectors is assigned a particular input function:

Selector 0	Circuit Board Integrity
Selector 1	Connector Integrity
Selector 2	Switch Position Status
Selector 3	Breaker Switch Status
Selector 4	CPU Status

The remaining data selectors read miscellaneous inputs that cannot be classified as any one type.

This scheme allows the programmer to write efficient supporting speech routines which run on CPU #5. The Scheduler polls the various data selectors using the X register as an index. Any abnormal condition is signified by a logic *high*, whereupon the Scheduler takes appropriate action, after which the contents of the X register are passed to CPU #5. The upper nibble (which determines the low-level selector being polled) is used to create a jump vector to a table of speech routines stored in ROM. The lower nibble tells CPU #5 which device of that particular type was in error; its identifier will be inserted into the blanks shown in the following speech routine examples:

0 *Interface circuit number __ not available. Please re-insert.*

1 *Connector __ is removed. Please replace.*

2 *Error! Check and reset switch number __.*

A logic high when polling the diagnostic network with X equal to the hexadecimal value of 24 would result in the speech output, *Error! Check and reset switch number 4.* In addition, the Scheduler would modify operations until the error was corrected. If a critical fault is not remedied by the operator after three such verbal requests, the Scheduler will shut down all secondary systems.

Referring now to figure 45, the following self-diagnostic scenario is given as means of illustration, using once again as an example the robot's propulsion system.

The Scheduler, upon receipt of movement instructions from the Planner, issues a *Move* command to CPU #4, the drive controller. After a brief wait to allow the drive motors to respond, the Scheduler then checks the left and right shaft encoder outputs (A) to ensure the motors are in fact turning, and if so, the diagnostic check stops here.

In the event no motion is detected, the Scheduler alerts any observers a problem has been detected by commanding CPU #5 to verbally announce the word *Error!*, and initiates the diagnostic procedure. The first check to be performed involves reading the battery voltage (B); if found to be low, the actual value is announced via the speech synthesizer, a request for operator assistance is made, and the system shuts down. This situation normally does not occur, as the robot continuously monitors supply status, and will initiate a recharging sequence when a low-battery condition is detected. It is possible, however, for the battery condition to deteriorate to an unacceptable level (precluding motion) before the recharge sequence is completed.

If battery charge is sufficient, the Scheduler next checks to see if a floor discontinuity has been detected (C), and if so, verbally announces its intentions, re-enables drive motion, and initiates a recovery operation based on the collective inputs of all six floor sensors. The location of the danger zone is entered into the model so as to avoid future occurrence.

Assuming no floor discontinuities, the *Emergency Abort* status is determined by reading the state of the flip-flop (D) (figure 45), and the operator is advised to reset if disabled. The robot, for safety reasons, cannot override the Emergency Stop system. If a fault still exists, the PWM circuit card integrity (E) is checked, followed by the ribbon cable (F) which connects the PWM card to the drive servo-amplifier in the base unit. Operator assistance is requested through speech synthesis if either prove to be in error.

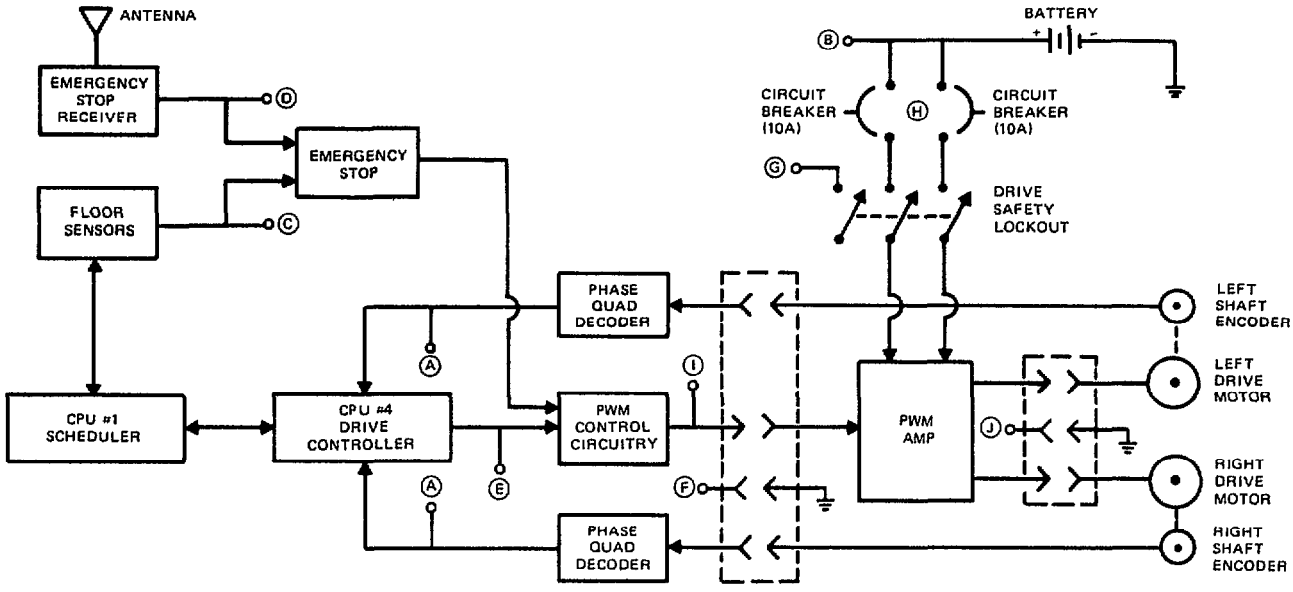


Figure 45. Revised drive system block diagram showing diagnostic circuitry checkpoints as discussed in the text.

The next check involves the Drive Safety Switch status (G), which if left in the off position would preclude motion. The associated speech routine is as follows: *Unable to move. Power disabled. Please restore.* This is followed by a check of the left and right circuit breakers (H). Continuing in a logical sequence, the Scheduler would check to see that PWM pulses are arriving at the servo-amplifier input (I), followed by a check of connector integrity (J) between the servo-amp unit and the motors themselves. The location of the fault, if found, is verbally announced; otherwise the Scheduler will secure the affected system and request assistance.

4.0 INTELLIGENT NAVIGATION

The ultimate goal of a robotic system is of course to perform some useful function in place of its human counterpart. Benefits typically associated with the installation of fixed-location industrial robots are improved effectiveness, higher quality, reductions in manpower, greater efficiency, reliability, and cost savings. Additional drivers include the ability to perform tasks for which humans are incapable, and the removal of humans from dangerous or life-threatening scenarios. The concept of mobility has always suggested an additional range of applications beyond that of the typical factory floor, wherein free-roaming robots move about with an added versatility that brought even greater returns. In practice, however, the realization of this dream has been slow in coming.

One of the most significant technological hurdles impeding the widespread introduction of mobile robotic systems arises from the need for a mobile platform to successfully interact with the physical objects and entities in its environment. The robot must be able to navigate from a known position to a desired new location and orientation, at the same time avoiding any contact with fixed or moving objects while en-route. There has been quite a tendency in many cases to oversimplify these issues in addressing various potential applications, and assume the natural growth of technology will provide the needed solutions. While this ultimately will come to pass, it is important to pace the development of the platform with a parallel development of the needed collision avoidance and navigational technology.

From a historical perspective there has been an evolution of mobile robot development that can be subdivided for purposes of this discussion into the two fundamental categories of teleoperated control and autonomous control, which will be discussed below.

The first attempts at implementing mobile systems were based upon the concept of teleoperation, initially referred to more commonly as remote control. The moving platform was steered by a human operator at a control station some distance away, based in some instances on visual feedback from an onboard television camera (a technique which is now referred to as *inside-out control*). More simplistic implementations involved control through observation of the device itself by an operator within visual range (now termed *outside-in control*). Two classes of teleoperated vehicles can be considered: (1) tethered vehicles, which employ some type of umbilical cable over which they receive control instructions and relay telemetry data, and (2) untethered, which are not connected by cable, but communicate with the control station via a radio link or other similar means. Primary advantages of teleoperation (from a navigational perspective) are the ability to travel anywhere as guided by the operator within the capabilities of the robot's propulsion and communications systems, and the ability of the system (by virtue of the operator in the loop) to make intelligent assessments and deal with unexpected conditions. Disadvantages include the tedious control demands placed upon the operator, restrictions in vehicle motion imposed by the length (weight) of the umbilical or range limitations of the RF system, and bandwidth limitations in the communications scheme.

The simplest form of autonomous control is sometimes termed *guidepath* control (figure 46), and involves a navigational control loop which reacts (in a reflexive manner) to the sensed position of some external guiding reference. The intent was to free the human operator from the requirement to steer the moving platform. Automated guided vehicles (AGVs) have found extensive use in factories and warehouses for material transfer, in modern office scenarios for material and mail pickup and delivery, and in hospitals for delivery of meals and supplies to nursing stations, to name but a few.

Advantages of *guidepath* control are seen primarily in the improved efficiency and reduction of manpower which arises from the fact that an operator is no longer required to guide the vehicle. Large numbers of AGVs can operate simultaneously in a plant or warehouse without getting lost or disoriented, scheduled and controlled by a central computer which monitors overall system operation and vehicle flow. Communication with individual vehicles can be over RF links or directional near-infrared modulated light beams, or other means. The fundamental disadvantage of *guidepath* control is the lack of flexibility in the system; a vehicle cannot be commanded to go to a new location unless the guide path is first modified. This is a significant factor in the event of changes to product flow lines in assembly plants, or in the case of a security robot which must investigate a potential break-in at a designated remote location.

Truly autonomous control implies the ability of a free-roaming platform to travel anywhere so desired, subject to nominal considerations of terrain traversibility. Many potential applications await an indoor mobile robot that could move in a purposeful fashion from room to room without following a set guidepath, with the intelligence to avoid objects and if necessary choose alternative routes of its own planning. But apart from the field of AGVs, examination will show successful implementation of robotic technology to date has been almost exclusively limited to fixed-place industrial robots operating in high-volume manufacturing scenarios justifying the intense *teach pendant* programming required to train the robot, which then repeats the taught sequences over and over under tightly controlled, highly structured conditions. The increasing use of process control and feedback sensors has started a definite trend toward adaptive control in flexible automation, to be sure.

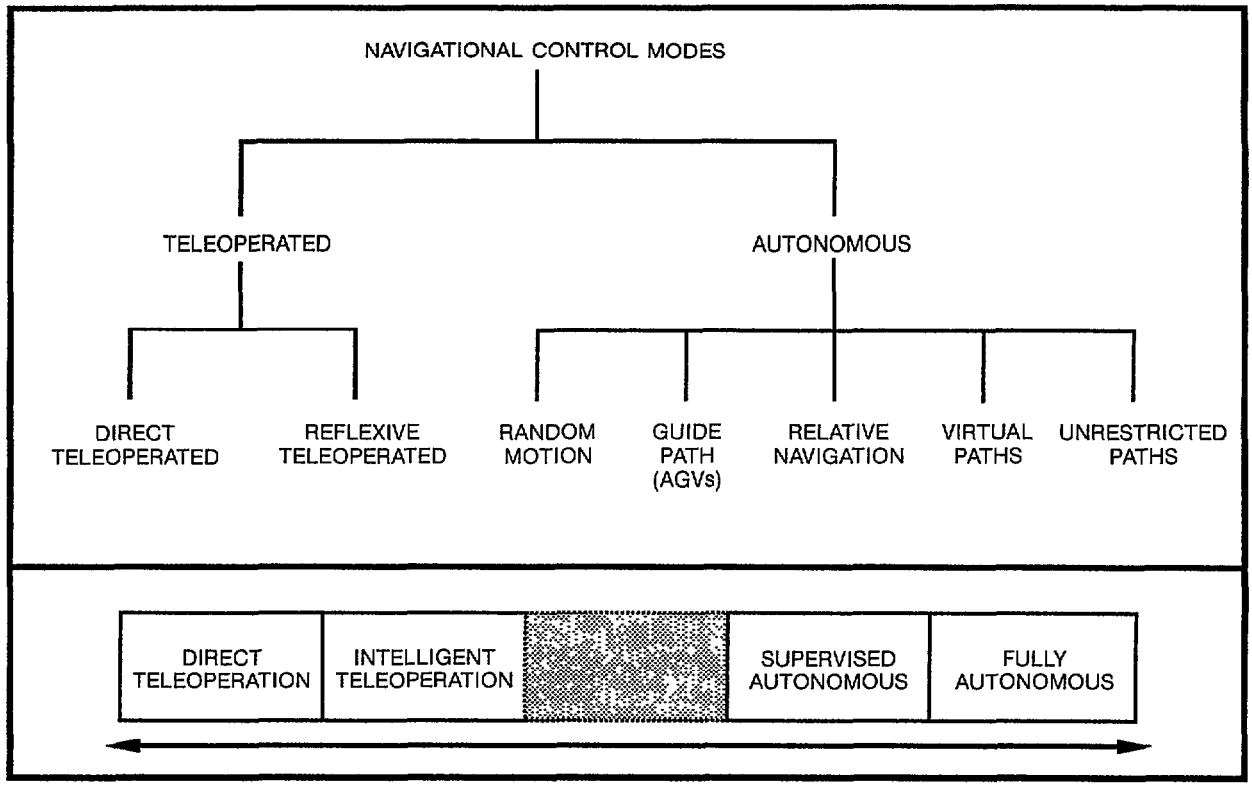


Figure 46. Taxonomy of navigational control modes employed on mobile robotic platforms. Degree of human involvement generally diminishes from left to right across the diagram.

Trying to directly carry this specialized assembly-line technology over into the unstructured world of a mobile robot, however, has met with little success; the problems are fundamentally different.

The difficulties can be directly related to the unstructured nature of the mobile robot's operating environment. Industrial process control systems used in flexible manufacturing (factory of the future) scenarios rely on carefully placed sensors which exploit the target characteristics. Background conditions are arranged to provide minimal interference, and often aid in the detection process by increasing the on-off differential or contrast. In addressing the collision avoidance needs of a mobile robot, however, the nature and orientation of the target surface is not known with any certainty; the system must be able to detect a wide variety of surfaces with varying angles of incidence. Control of background and ambient conditions may not be possible. Preprogrammed information regarding the relative positions, orientations, and nature of objects within the field-of-view of the sensors becomes difficult indeed for a moving platform.

Specialized sensors specifically intended to cope with these problems must be coupled with some type of *world modeling* capability representing the relative/absolute locations of objects detected by these sensors in order to provide a mobile platform with sufficient awareness of its surroundings to allow it to move about in a realistic fashion. The accuracy of this model, which is constructed and refined in a continuous fashion as the robot moves about its workspace, is directly dependent throughout this process upon the validity of the robot's perceived location and orientation. Accumulated dead-reckoning errors soon render the information entered into the model invalid in that the associated geographical reference point for data acquired relative to the robot's position is incorrect. As the accuracy of the model degrades, the ability of the robot to successfully navigate and avoid collisions

diminishes rapidly, until it fails altogether. A robust navigational scheme that preserves the validity of the world model for free-roaming platforms has remained an elusive research goal, and for this reason many proposed applications of autonomous mobile robots are yet to be implemented.

In summary, navigational schemes for mobile platforms can be broken into the two broad categories of teleoperated control and autonomous control, as shown in figure 46. Autonomous control can be further broken down into guidepath control, and unrestricted path (world model) control. Each of these as they relate to this effort will be addressed in some detail in the following sections, leading up to a discussion of a NOSC-developed hybrid navigational control scheme which addresses the problem of world model degradation due to cumulative dead reckoning errors.

4.1 REFLEXIVE TELEOPERATED CONTROL

Remotely operated platforms have been employed for a variety of applications dating back to World War II. Recent advances in the applicable technologies have brought about an even greater interest in such systems for use in hazardous scenarios, such as the nuclear power industry, underwater search and recovery, firefighting, and bomb disposal, to name but a few. Often, however, the advantages afforded by removing the operator from a dangerous situation are significantly offset by the difficulties encountered in controlling the remote vehicle, thus limiting the number of practical applications to those for which the perils associated with direct exposure justify the alternative. One of the more fundamental issues in this regard involves controlling vehicle motion based on visual feedback provided to the operator by an onboard camera; loss of depth perception and visual acuity are perhaps the most important factors influencing the resultant degradation in performance.

Much work has been done over the past decade to improve the man-machine interface issues for teleoperated systems in an attempt to achieve a higher operational equivalence ratio (loosely defined as the amount of time it takes to perform a series of tasks, divided by the amount of time it would take an operator to perform the same tasks remotely). The concept of remote telepresence introduces stereo vision as well as binaural hearing to give the operator a stronger sense of being in the working environment. Head-coupled displays have been employed to further enhance this feeling, wherein a pair of high-resolution cameras on the remote (slave) end are positioned in realtime in accordance with the pan and tilt movements of a helmet worn by the operator at the (master) control station (Martin and Hutchinson, 1989). The helmet is equipped with miniature monitors to relay the video output from the left and right cameras to the left and right eyes of the operator, thus providing some degree of depth perception. As the operator turns his head toward elements of interest in the scene being viewed, the remote slave positions the cameras accordingly to achieve the desired input. While the 3-D capability thus provided is a decided improvement over monocular vision, the negative side effects include extremely high operator fatigue, a tendency to induce nausea in some operators, higher bandwidth requirements in the control loop, and significantly enhanced system complexity.

Reflexive teleoperation, therefore, is intended to provide a robust means for remote operation of an unmanned vehicle, which frees the operator from the lower level concerns associated with direct teleoperation: speed of the vehicle and vehicle direction are servo-controlled by an onboard processor as a function of the surrounding environment in response to local sensor inputs, but under the high-level supervisory control of the remote operator.

A good analogy here might be to consider a person riding a motorcycle as compared to someone riding a horse. A motorcycle has absolutely no inherent intelligence; the driver must

attend to every detail regarding steering and velocity of travel, or suffer the consequences. A horse, on the other hand, can be directed to follow an equivalent route by periodic *high level* prompts; the rider then supervises the resulting actions, making corrections as necessary if the horse deviates from the intended path. The point to note, however, is the horse can negotiate on its own an opening between trees, for example, and is smart enough not to run into surrounding obstacles, or fall into a ditch. The reflexive teleoperation scheme attempts to emulate this type of control mode on the remote platform, by providing some degree of local intelligence which seeks out openings between obstructions. This virtually eliminates the possibility of collisions in cluttered surroundings, while significantly reducing the stress and fatigue associated with conventional remote operation of a mobile system.

The console operator can effect a hand-off from autonomous to teleoperated control by halting the robot and calling up the *Teleoperated Mode* on the system menu. The onboard dead-reckoning software keeps track of the robot's position and heading while executing drive commands directly from the control console. This mode could be used to explore some new area (mapping the surroundings in the process if so desired), or to move expeditiously to a new location, or around a perceived threat which may not be obvious to the robot. Upon completion of this action, the operator relinquishes control back to the path planner, and autonomous operation resumes.

As discussed in section 3.3, ROBART II is equipped with a number of noncontact ranging and proximity sensors which are used to determine the relative positions of surrounding objects. A combination of ultrasonic rangefinders and near-infrared proximity detectors are employed to increase the chances of obstacle detection beyond that which would be realized with either system alone. Interaction of the emitted energy with the target surface causes each type to have its strengths and weaknesses, and accordingly, each can detect certain targets which the other can't. As an example, the shorter wavelength of near-infrared energy greatly reduces problems associated with specular reflection, and as a result, significantly improves chances of detecting a sharply-angled target surface which the ultrasonic sensor may miss entirely. On the other hand, near-infrared energy will pass right through a glass door or window (which subsequently goes undetected), while ultrasonic energy will be reflected.

A good illustration of this latter situation is seen when a photographer tries to take a picture through a glass window using an autofocus camera. If the autofocus system employs ultrasonic rangefinding, the camera will focus on the window itself, which reflects the ultrasonic waveform. If a near-infrared triangulation ranging system is employed, then the camera will focus on the scene outside the window, since the near-infrared energy passes through the glass with little reflection. The use of both near-infrared and ultrasonic systems therefore greatly improves the overall probability of detection of surrounding obstacles. A system block diagram is presented in figure 47.

In the *Teleoperated Mode*, commands are entered by the operator at the host computer, and received by the remote system over the serial RF data link, thus establishing the vehicle's commanded heading and velocity. Referring now to figure 48, the center key ('5') of the numeric keypad is the *Halt* key; which immediately stops any vehicle motion. The Up Arrow ('8') is used to request forward motion; each time the key is depressed, the commanded forward speed is incremented, up to a maximum value. The Down Arrow ('2') is used to request reverse motion if the vehicle is stopped, or to decrement the forward speed value if moving forward. Similarly, pressing the UP Arrow will decrement the commanded speed if the vehicle is moving backward. The Left Arrow ('4') will cause the vehicle to rotate in place to the left if stopped, just as the Right Arrow ('6') will cause in-place rotation to the right.

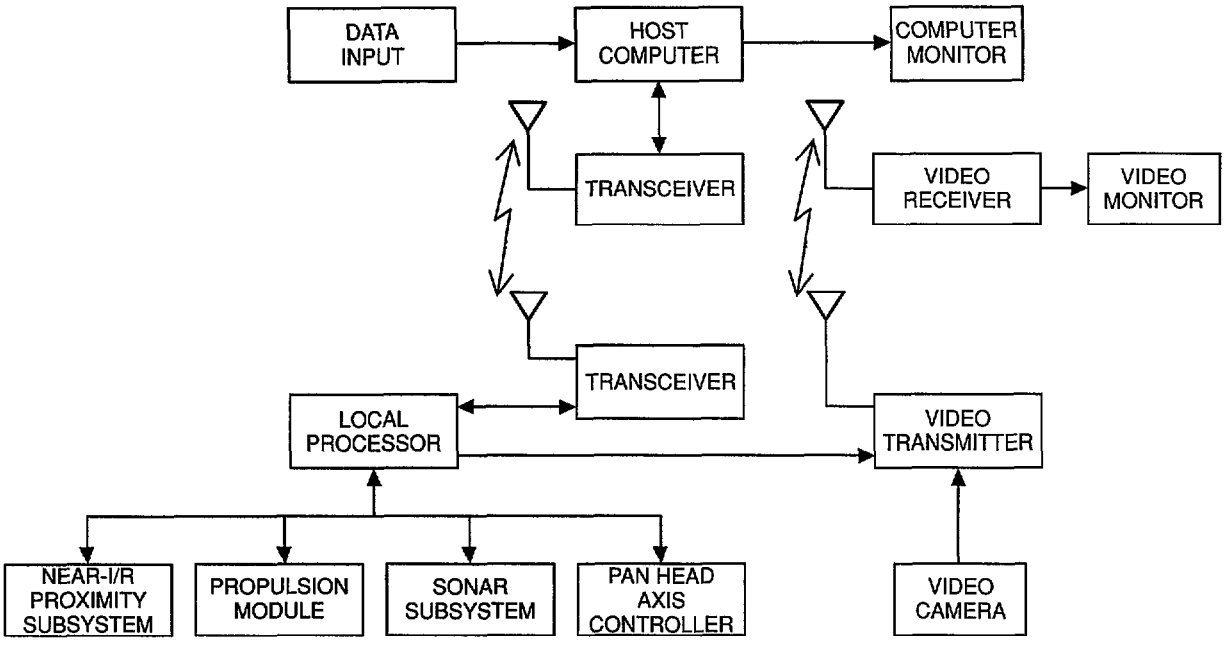


Figure 47. System block diagram illustrating the various subcomponents employed in the reflexive teleoperated control scheme.

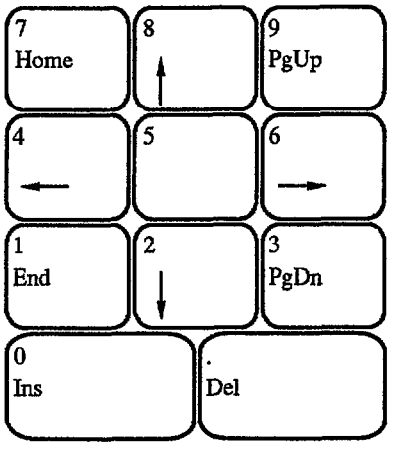


Figure 48. Layout of the numeric keypad of the host computer used by the operator to enter supervisory motion control commands when in the teleoperate mode.

If the robot is moving forward (or reverse), however, the left and right arrow keys superimpose a turn offset onto the drive command. As discussed in section 3.2, the platform employs differential steering, with 15 possible speed commands for both the left and right drive motors. Velocity commands are of the form of a single hexadecimal byte, with the upper nibble specifying velocity for the left motor, and the lower nibble specifying for the right. For a baseline velocity command of $55 (both motors running at speed 5), a single stroke of the Left Arrow key would result in a commanded velocity of $46 (left motor speed 4, right motor speed 6). Pressing the key again would result in $37, for a sharper turn, and so forth.

The Scheduler evaluates the surrounding environment in conjunction with the ordered

commands from the control station in order to determine the actual direction of travel, as well as the maximum safe speed, not to exceed the commanded velocity. The necessary control functions can thus be subdivided into the two tasks of establishing the appropriate parameters for vehicle speed and direction, which will be discussed separately in the following sections.

4.1.1 Speed Governing

The maximum safe speed for the remote vehicle is derived from the minimum range returned by the sonar ranging system in the direction of intended motion, using the center three transducers in the lower array depicted in figure 8, in accordance with the following rules:

> Any range reading (from these three transducers) which is less than a specified minimum threshold (i.e., three feet) is treated as the minimum range, regardless of vehicle direction.
>
> If the vehicle is going straight, the returned range from the center transducer is used as the minimum range if no transducer readings fall below the above minimum threshold.
>
> If the vehicle is turning to the right, the lesser value of the returned ranges from the center and right transducers is used as the minimum range.
>
> If the vehicle is turning to the left, the lesser value of the returned ranges from the center and left transducers is used as the minimum range.

In this way the detection zone is shaped to conform to the area of interest, and the maximum safe speed can be calculated based on the perceived congestion within the area of intended transit, simply by scaling the determined minimum range value. The maximum safe speed is therefore linearly proportional to the unobstructed ultrasonic range value returned in the direction of vehicle motion.

The software for reflexive teleoperation is set up as a loop which calls the governing routine to establish the maximum safe speed, and then looks for potential obstructions, as will be discussed in section 4.1.2 below. Each time the governing loop is called, the appropriate sonar readings are analyzed to generate the maximum safe speed, and this speed is requested from the drive controller in accordance with the following rules:

> If the calculated maximum safe speed is less than or equal to the commanded speed (from the operator), the maximum safe speed is to be ordered (passed to drive controller).
>
> If the the maximum safe speed is greater than the commanded speed, then the commanded speed is to be ordered.
>
> If the current speed is greater than the speed to be ordered, then the new (slower) speed is ordered immediately.
>
> If the current speed is less than the speed to be ordered, then the current speed is incremented.
>
> If the current speed equals the speed to be ordered, then no action takes place.

The first two rules ensure the robot does not go faster than requested by the operator, nor faster than deemed safe for current conditions. The last three rules cause the drive controller to immediately slow down to an ordered slower speed, but to successively ramp up to an ordered faster speed. Speed governing can be disabled by the operator if so desired by pressing the character 'E' (Emergency Override) on the keyboard, whereupon the vehicle baseline velocity is as commanded, with no regard for surrounding obstructions.

4.1.2 Directional Control

Actual direction of travel is based on the commanded rate of turn modified as appropriate by information derived from both the ultrasonic rangefinders and the near-infrared proximity sensors, which collectively form a protected area in front of the robot with zones of coverage as shown in figure 49. The sensor outputs are analyzed to identify which of the following cases applies:

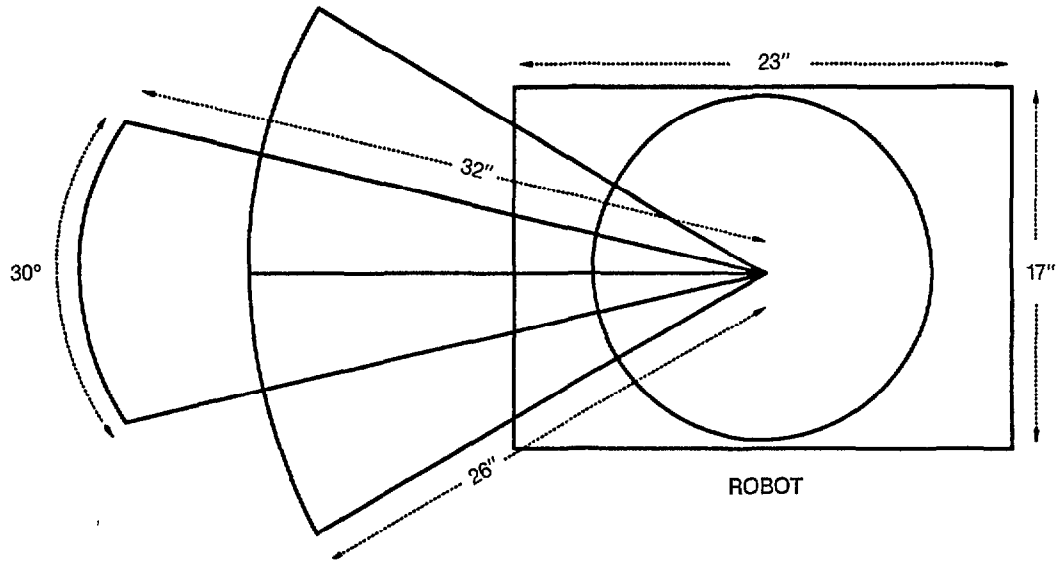


Figure 49. Center, left, and right zones of coverage for the near-infrared proximity and ultrasonic ranging sensors.

(a) No Obstruction

(b) Obstruction Left

(c) Obstruction Right

(d) Obstruction Center

(e) Blockage

In case *a* above, the commanded rate of turn is unmodified, and the vehicle moves in direct response to the operator's input at the control console. In case *b*, the onboard microprocessor would alter the drive command so as to cause the vehicle to turn right, away from the obstruction. Similarly, in case *c*, the vehicle would veer to the left. Note cases *b* or *c* can occur in conjunction with case *d*, whereupon the offset correction is increased, so as to provide a faster evasive rate of turn.

Case *d* by itself represents a scenario where an obstruction appears directly in front of the vehicle, but is not seen by either of the side sensors. This situation is encouraged, to some extent, by the fact that the center proximity sensor is adjusted to have a maximum detection range of 32 inches, whereas the side-looking sensors are set for 26 inches. (The ultrasonic ranging sensors are similarly weighted.) The special treatment of case d represents one of the key elements of the reflexive teleoperation control scheme. The correct avoidance maneuver for this situation involves the vehicle turning one way or the other until a clear area for transit appears; the direction of turn could be random, as initial sensor readings favor neither left nor right, but this results in somewhat unpredictable motion.

Instead, the desired direction of turn is determined in advance by the operator, using the '7' and '9' keys of the numeric keypad to specify 'left' and 'right', respectively. (Pressing either key will activate the *Reflexive Mode*, which can be deactivated by pressing the '0' key.) In this way, the operator preselects the direction to turn in the event the path becomes blocked, a feature which can be used to advantage in driving the vehicle. Note as the vehicle

begins to execute this turn in response to case d, the sensor input will change accordingly, usually to some combination of case *d* in conjunction with either case *b* or *c*, depending on the direction of turn. This new situation then evokes a continuing turn in the same direction as previously discussed. As this avoidance maneuver continues, the center sensor should eventually clear, and the rate of turn will be decreased, until such time as the side sensor clears, whereupon the vehicle resumes its commanded travel.

For example, in the map illustrated in figure 50, a preselected right turn ('9') entered anywhere after point A on path segment ABC will cause the platform to turn right at point C when exiting room 103 and entering the hallway, after which the system will proceed to point D. Note, however, the slight course correction to the left at point B (case c), due to the presence of the cabinet to the right. Similarly, a preselected left turn along ABC will cause the vehicle to go to destination F. In the event of a right turn at C, operator intervention at point D in the form of a right turn command ('6' or Right Arrow) will cause the system to turn and enter the open bay area; otherwise the vehicle would continue to point E.

Case *e* is any combination which includes cases *b* and *c* together, which corresponds to a situation where the robot's path is blocked with some degree of symmetry relative to the intended path, and simply veering in the direction of free space is not possible. In other words, the path is sufficiently blocked that the vehicle must turn in place until a clear exit path appears. The direction in which to pivot is determined by the prespecified turn preference. For instance, as the robot travels from point D past point E in figure 50, it comes to the end of the hallway, which ultimately will evoke a case e response (the doorway to room 107, a utility closet, is always kept shut). The robot will then rotate in place (to the right, unless the preselected direction was changed by the operator somewhere along path segment CDE) until it is facing the open hallway looking back towards point D. Forward motion is then resumed along path segment ED.

4.1.3 Seek Mode

The preceding discussion relates for the most part to scenarios where the robot is moving towards some desired opening, such as a doorway, which is generally in the vicinity of forward travel. However, another situation that arises quite frequently in normal operations involves an opening off to the side of (yet in close proximity to) the path of travel. A good illustration of this situation is seen in the case where the robot is being driven down a narrow hallway, with the desired intention of entering a side door opening onto that hallway, such as illustrated by a left turn into room 105 from path segment FCDE in figure 50. The problem becomes one of perspective, in that the forward-looking sonar and proximity sensors which provide reflexive control are situated so as not to be able to detect the opening to the side. Similarly, the doorway will pass from the video camera field-of-view before the robot has arrived at the proper point to initiate a turn, and so the operator must judge from prior experience (or pan the head accordingly) to effect the proper maneuver. In some cases, the doorway may never appear in the video scene, due to the angle or route of approach.

In any event, it would be nice to be able to download the entire problem to local intelligence and sensors onboard the vehicle, in keeping with the concepts of reflexive teleoperation introduced in the preceding sections, and this is the intent of the *Seek Mode* which is the subject of this section. The problem is subdivided as follows:

Activating *Seek Mode*: the robot must be instructed to look for an opening on the left or right side.

Automatically providing the operator with visual feedback.

Locating the actual opening.

Initiating the turn into the opening at the proper time.

Negotiating the opening itself.

Terminating the *Seek* action and resuming normal transit.

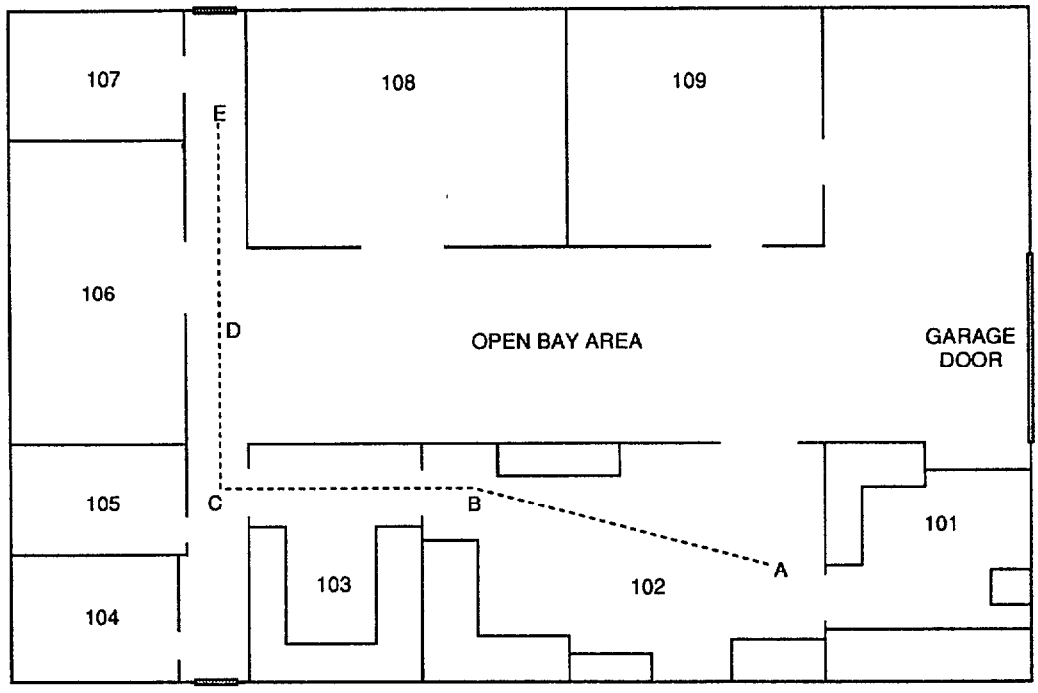


Figure 50. Floor map of Building F-36 illustrating route options under reflexive teleoperated control.

The first task is accomplished through use of the two remaining unused keys in the numeric keypad at the operator control station: depressing the '1' key initiates a seek action on the left side, and depressing the '3' initiates seek action on the right. Immediately upon receipt of the subsequently generated *seek* command, the Scheduler turns the robot's head 80 degrees in the direction requested, and activates the programmable near-infrared proximity detector (section 3.3.4) in such a way as to indicate any reflective targets within 5 feet. The head-mounted surveillance camera is now positioned so as to provide the operator with a view of the upcoming doorway (figure 51) as the robot continues moving along its original course, thus satisfying the second requirement in the list above.

Identification of the doorway is accomplished by monitoring the output of the programmable proximity sensor, with appropriate filtering to account for minor deviations due to specular reflection. The proximity sensor must indicate a target (wall) present within 2 seconds of the head assuming the near-orthogonal seek position, and this target must remain valid for at least two more seconds. The Scheduler then begins looking for a *target-to-no-target* transition, and the *no-target* condition must persist for 1 second before turning is initiated. If any of these conditions are violated, or if a new command is sent down by the operator, the system kicks out of *Seek Mode* and returns to *Reflexive Mode*, centering the head in the process.

Once the doorway has been found, the Scheduler initiates a turn in the appropriate direction to align the intended path of travel with the identified opening. The turn continues until any of the forward-looking sonar or proximity sensors used for conventional reflexive control encounter a target, or until the robot has turned 90 degrees to the original heading. At this point, *Seek Mode* is deactivated by the Scheduler, the head pans to return the surveillance camera to the center position, and reflexive control takes over, guiding the robot through the doorway. For this reason, the left and right *Seek Mode* commands will automatically set the corresponding desired turn direction for normal reflexive mode, to ensure a smooth transition.

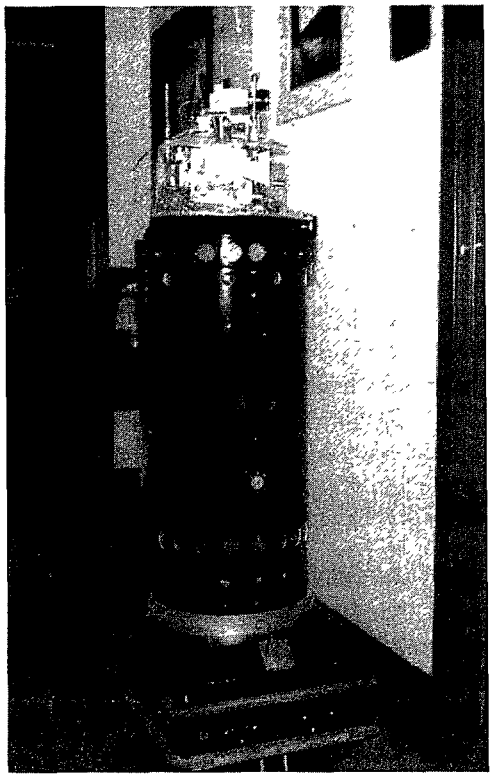


Figure 51. The head pans automatically in the specified *seek* direction as shown here for a left turn into room 105.

Commands are of the following format: 0WXX

where the leading '0' indicates the command is intended for the robot, and the 'W' instructs the Command Parser software routine that this command pertains to reflexive teleoperation. (See also appendix B. The 'W' is a holdover from the original 1983 implementation of the code running on the Scheduler, as a simple *Wander* routine that avoided obstacles, moving randomly in an exploratory fashion.) The single parameter *XX* is interpreted as shown below:

Key	XX	Meaning
0	00	reflexive and seek modes disabled
7	01	reflexive enabled, preselected left turn
9	02	reflexive enabled, preselected right turn
1	11	seek enabled, left side
3	22	seek enabled, right side

When *Seek Mode* is successfully terminated by the Scheduler, the upper nibble of the parameter *XX* is cleared, leaving the system in the reflexive mode with a predisposition to turn in the previously specified *seek* direction.

4.2 ABSOLUTE WORLD COORDINATE CONTROL

Providing an autonomous capability to support non-restricted intelligent movement involves the implementation of an appropriate map representation, the acquisition of information regarding ranges and bearings to nearby objects, and the subsequent interpretation of that data in building and maintaining the world model.

4.2.1 Selecting a Map Representation

The simplest map representation is a two-dimensional array of cells, where each cell in the array corresponds to a square of fixed size in the region being mapped. Free space is indicated with a cell value of zero; a nonzero cell value indicates an object. The simplest and most compact form of a cell map consists of one bit per cell, and thus indicates only the presence or absence of an object. By using multiple bits per cell, additional descriptive information can be represented in the map, to include the probability of a given square being occupied. This is useful when the precise location of an object is unknown. Memory usage is independent of map content, so cluttered surroundings are not a problem. However, the resolution of the map is only as good as the square size, and doubling the resolution quadruples the memory requirements.

A slightly more sophisticated and elegant approach is to use a quadtree representation (Fryxell, 1988). Each map begins as a square which is subdivided into four smaller squares. Each of these squares is in turn recursively subdivided (down to the map resolution if necessary) until the region occupied by the square is homogeneous (all object or all free space). For an uncluttered environment, a substantial savings in memory usage is achieved, with a decrease in find-path execution time, since the effective map size is smaller. In highly inhomogeneous environments however, memory usage can increase beyond that of the simple cell map,

thus negating the primary advantage of the quadtree. Octrees can be used if a three-dimensional representation is required.

A third technique uses polyhedra and curved surfaces or geometric primitives to represent objects in the workspace (Lozano-Perez and Wesley, 1979; Brooks and Lozano-Perez, 1983). Such maps are quite compact, and with no inherent *grid*, the locations of the objects can be precisely entered into the model. These maps are also easily extended into three dimensions, in contrast to the cell map where memory cost would be prohibitive. However, updating the map with real-world data is difficult, as it is hard to accurately glean polygonal information from inexpensive sensors mounted on a mobile robot. Statistical uncertainty of the existence of objects is difficult to implement as well.

Brooks has devised a scheme based on generalized cones (Brooks, 1983). Rather than map both obstacles and free-space, only the areas the robot can freely traverse are mapped, and the robot is required to stay within these *freeways*. Paths can be quickly found using this method and do not *hug* obstacles as many other algorithms do, thus decreasing the likelihood of a collision due to cumulative dead reckoning errors. However, this algorithm does not work well in cluttered environments, and decomposing free space into generalized cones can be computationally expensive.

Following an examination of these and various other alternatives, a cell-based map was adopted for a number of reasons:

A three-dimensional map was thought unnecessary as the primary concern was navigation, not the manipulation of objects in 3-space. Each object is treated simply as its projection on the X-Y plane.

The area described by the map is a small bounded space, (i.e., a building interior), where a relatively coarse grid (3-inch resolution) can be used.

Objects of unknown configuration are easily added. This feature is of particular importance since low-resolution ultrasonic sensors are used for map updating.

The traversibility of a square can be statistically represented and easily changed.

A simple Lee maze router (Lee, 1961) can be used for path planning.

The map can be accessed and updated quickly, which is extremely important for realtime operation.

4.2.1.1 Pan and Zoom Capability. One of the undesirable side effects of using a bit-mapped world model is that it is difficult to scale the image to optimally fit within in the display area of the monitor. Conventional integer scaling techniques typically yield quick screen updates, but may result in large maps being either too small to be useful or else not fitting on the display (figure 52). Floating point and image processing techniques can be used to make the map fit neatly on the screen, but with a substantial reduction in the update rate. One way around these problems is to use integer scaling augmented with zooming and panning functions.

When the robot is known to be operating only within a small subset of a large map, the user can elect to zoom in on the area of interest. This is done by first selecting the *Zoom In* function of the *Zoom* pulldown menu. The user can then use the mouse to mark on the map two corners of a bounding box containing the desired area (figure 53). The map is then rescaled with the marked area blown up to fit within the screen limits (figure 54). If the user wishes to see more of the map, he can select either *Zoom All* or *Zoom Out*. The *Zoom All* function shows the entire map, while *Zoom Out* increases the displayed area by 25 percent.

A panning function has also been implemented to facilitate the display of large maps. To view other portions of the map, the user can use the arrow keys to *jump scroll* the map in the appropriate direction. The map pans by

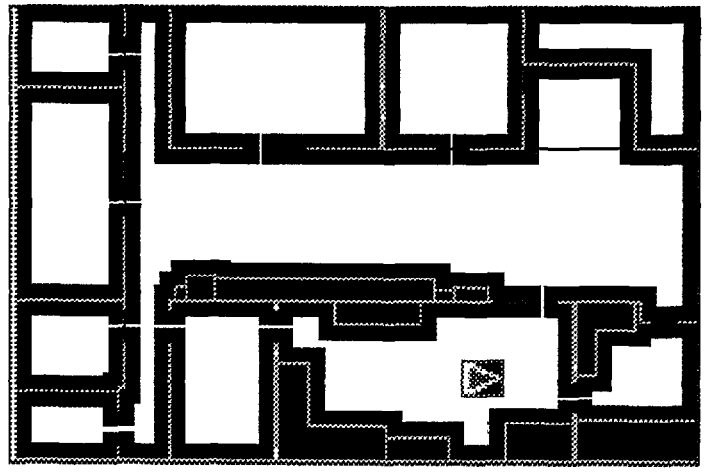


Figure 52. Large maps may appear too small when integer-scaled to fit the display.

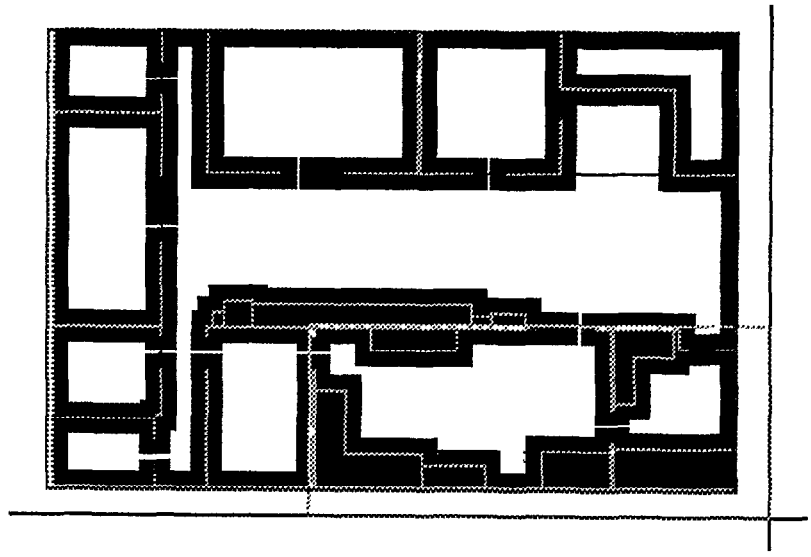


Figure 53. The mouse is used to mark two corners of a bounding box which determines the area to be displayed when zooming.

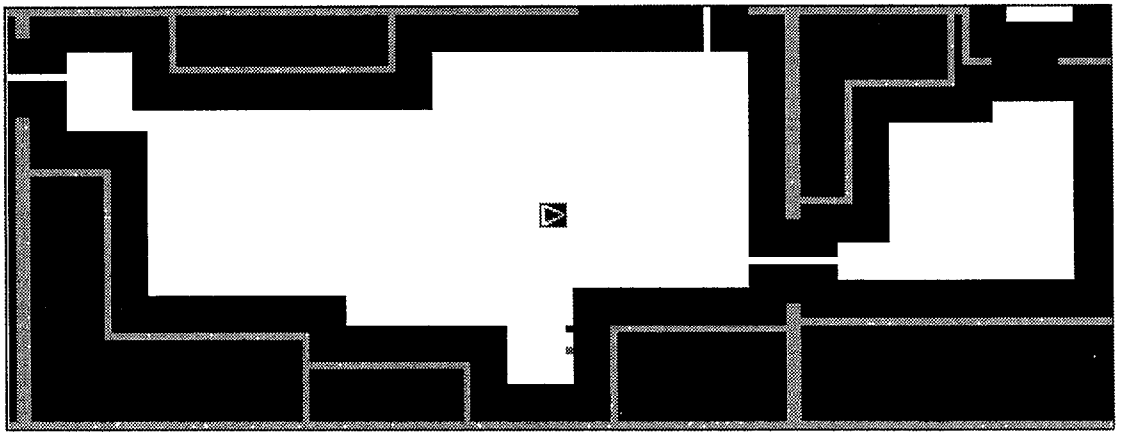


Figure 54. Resulting zoomed display from example depicted in figure 53.

about 25 percent of the portion displayed in the direction of the arrow. If the user holds down the shift key while hitting the arrow, the map scrolls by only one square for even finer control.

An *autopan* feature is invoked when operating in the autonomous modes. When the user is specifying a path destination with the mouse, the screen automatically pans as needed in the direction of cursor movement. Similarly, during execution of a path, the screen automatically pans when the robot moves off the edge of the current map, recentering the robot on the new displayed area.

4.2.1.2 AutoCad Translation. Constructing a map from the ground up using the Map Editor can be tedious and time consuming. Often, however, a CAD drawing of the building has already been created. A translator has therefore been written to convert an AutoCad Drawing Interchange File into a bit-mapped model that the path planner can use. The only user input required is the desired square size. Items such as doors, recharging stations and recalibration positions will need to be entered after the conversion using the Map Editor as before, but all semipermanent entities such as structural walls and furniture are taken care of automatically.

4.2.2 Acquiring and Incorporating the Data

The acquisition of range data by both the navigational and collision avoidance sonar arrays is initially addressed in section 3.4. Such target distance information must next be entered into the world model as the robot is moving. This seemingly trivial operation turns out to be somewhat difficult due to problems associated with the operation of ultrasonic ranging systems in air. These include temperature dependence, which has a significant impact on range accuracy, and beam dispersion, which contributes to angular uncertainty. Specular reflections from target surfaces can cause additional problems, in that the wavelength of ultrasonic energy at the frequencies employed on the Polaroid ranging module is approximately a quarter of an inch.

Adjacent sensor interaction requires the transducers in the array be individually fired in sequence rather than simultaneously. Finally, the slow speed of sound in air results in marginal update rates, as well as the need for successive coordinate transformations to account for the displacement due to robot motion during the sequential firing of all transducers in the array (Everett, 1985).

Consequently, effective interpretation and use of range data is critical to achieve a reasonably accurate representation of surrounding obstacles. Moravec and Elfes (1985) of CMU describe a scheme for mapping sonar range returns using probability distribution functions. For each sensor reading, the assigned probability of an object being at the exact indicated range and bearing is high, and decreases radially around that point according to a specified distribution function. In addition, a second distribution function characterizes the *emptiness* of cells between the sensor and the returned range. Points near the sensor have a high probability of being unoccupied, with decreasing probability for those points closer to the indicated range and off the beam axis.

The CMU technique is applied to a map where the state of occupancy for all cells is initially marked as unknown. The robot is moved to various vantage points in the room; several sonar readings are taken at each point, and averaged to create the probability map. The robot thus creates its own map in an exploratory fashion, but must stop periodically to take sensor readings.

Fryxell (1988) and Beckerman and Oblow (1988) also use probability schemes for mapping sonar data. Fryxell takes sonar readings (modeled as rays) from different places in the environment and constructs two arrays, one observing the number of times each cell was *hit*, and the other observing each time a cell was *missed*. A *voting* procedure combining both maps is then used to create the final map, where each cell is marked as either occupied or unoccupied.

Beckerman and Oblow (1988) use a similar method, but model the sonar beam as a cone subtending an angle of 18 degrees. The reduced effective beamwidth (18 degrees versus 30 degrees) is achieved by employing a phased array consisting of four transducers, the array being sequentially repositioned mechanically to achieve the desired coverage. As with Fryxell (1988), the robot was moved to various vantage points in the room to make sonar observations. These data are saved in auxiliary buffers and used to update a cumulative map, with each cell labeled as conflicting, unknown, occupied, or empty. (A conflicting cell occurs when one or more sonar readings intersect such that one marks the cell as occupied while the other marks it as empty). After all the nonconflicting data has been integrated into the cumulative map, the original saved data for each observation point are used to resolve the status of the remaining conflicting cells through pattern analysis. This technique generated maps similar to those created by Fryxell's method, but with better resolution even though fewer sonar readings were taken.

A faster and less computationally expensive variation of these procedures was implemented on ROBART II. By using a simplified probability scheme and range gating fixed arrays of sonar sensors, the mapping process can take place in realtime while the robot is in motion. Two different mapping procedures are used, one for creating the initial map, and another for updating the map during the execution of a path.

To generate the initial map, the robot moves very slowly around the room boundaries while firing all 24 transducers in the upper navigational array. The sonars are modeled as rays and the data is range-gated to 6 feet. If the indicated distance is less than 6 feet, the probability value assigned to the cell corresponding to that location in the map is incremented once. After the room had been completely traversed, the cell values are thresholded to remove noise and achieve the best map definition. The resulting map now contains the known *permanent* objects in the room as seen by the robot, and

can next be manually edited to add additional features, such as hidden lines, doorways, etc. Each object in the map is then automatically *grown* by half the width of the robot in order to model the robot as a dimensionless point during subsequent find-path operations (Lozano-Perez and Wesley, 1979). Permanent objects thus created under human supervision cannot be later erased by the robot during path execution.

When entering data from the collision avoidance sonars into the world model during actual execution of a path segment, a different scheme is used, with only the center five transducers in the lower array activated. If a given sensor return shows an object is within 5 feet, the cell at the indicated location is incremented twice (up to a specified maximum, currently 16). Also, the probability value assigned to each of the eight neighboring cells is incremented once, to partially take into account uncertainties arising from the 30-degree dispersion angle of the ultrasonic beam. Cells previously marked as being permanent objects or growth (see above), however, are left untouched since they will always be avoided by the path planning search.

In addition, each time a sonar range return is processed, all the cells within a cone 10 degrees wide and 4 feet long (or less if an object appears within 4 feet) have their assigned values decremented by one. This erodes objects no longer present, and also serves to refine the representation of objects as the robot approaches. Transient objects are erased from the map at a slower rate than they are entered, so the system tends to err on the side of not running into obstructions. As with object addition, permanent obstacles and growth are left untouched.

Figure 55 shows a three-dimensional bar chart depiction of such a map, where the height of each bar is proportional to the probability that the given cell is occupied. The probability cluster at point A marks the location of a chair which had been placed at the right-hand end of the room. The robot was told to execute a rectangular path, causing it to completely circumnavigate the chair. After the robot's first pass around the room, the chair was moved to point B and the path repeated, with the resulting probability distribution as shown in figure 56. Note the robot quickly recognizes the new location of the chair, while the probability of an object being present at point A has been significantly decreased. In addition, two additional objects can be seen at points C and D. After the third pass around the room the old representation of the chair has almost completely decayed (figure 57), and by the fourth pass, it is gone completely (figure 58). Objects which do not change position between passes are further reinforced until their associated probability functions saturate. The small cluster at point E in figure 58 resulted when an observer entered the room through the doorway, and then quickly exited upon realizing a test was in progress.

4.2.3 Operating on the Navigational Model

The world model (map) contains positional information about all the known objects in the environment, and may be either robot or human generated, or a combination of both. In either case, only relatively immobile (hence the term *permanent*) objects such as walls, desks, filing cabinets, etc., are recorded during the initial map generation procedure. Objects likely to be transitory in nature are not recorded (chairs, trash cans, carts, etc.), and present a problem during actual path execution, giving rise to the need for an effective collision avoidance capability.

> Find a path to the destination. If no path exists, then return a value of FALSE to the calling program.

4.2.3.1 Path Planning. There are four basic issues the path planning algorithm must solve to get the robot from point A to point B:

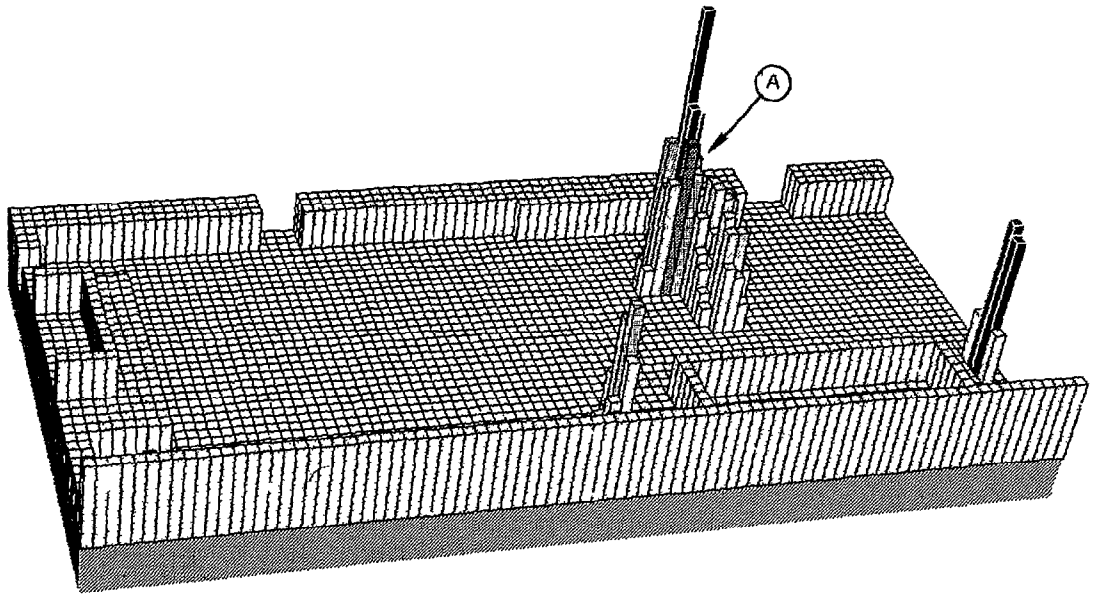


Figure 55. Three-dimensional probability distribution plot showing the perceived location of nearby objects in the environment. Note representation of chair at A.

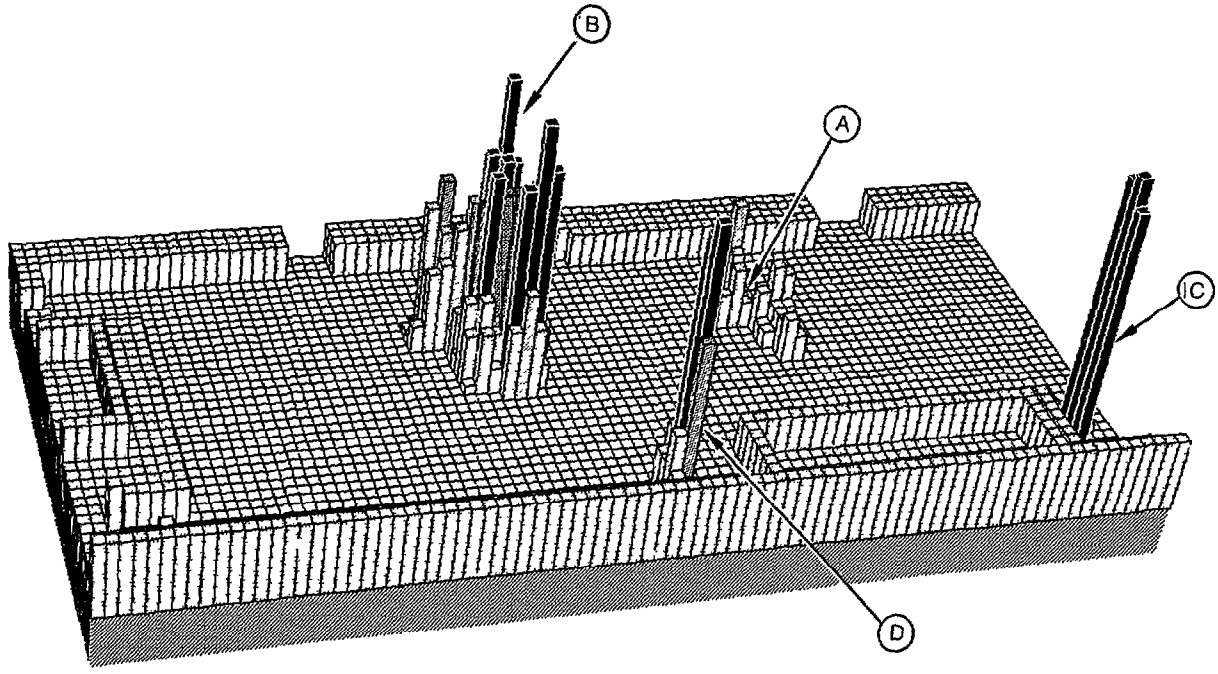


Figure 56. Probability distribution after moving chair from location A to new location at B.

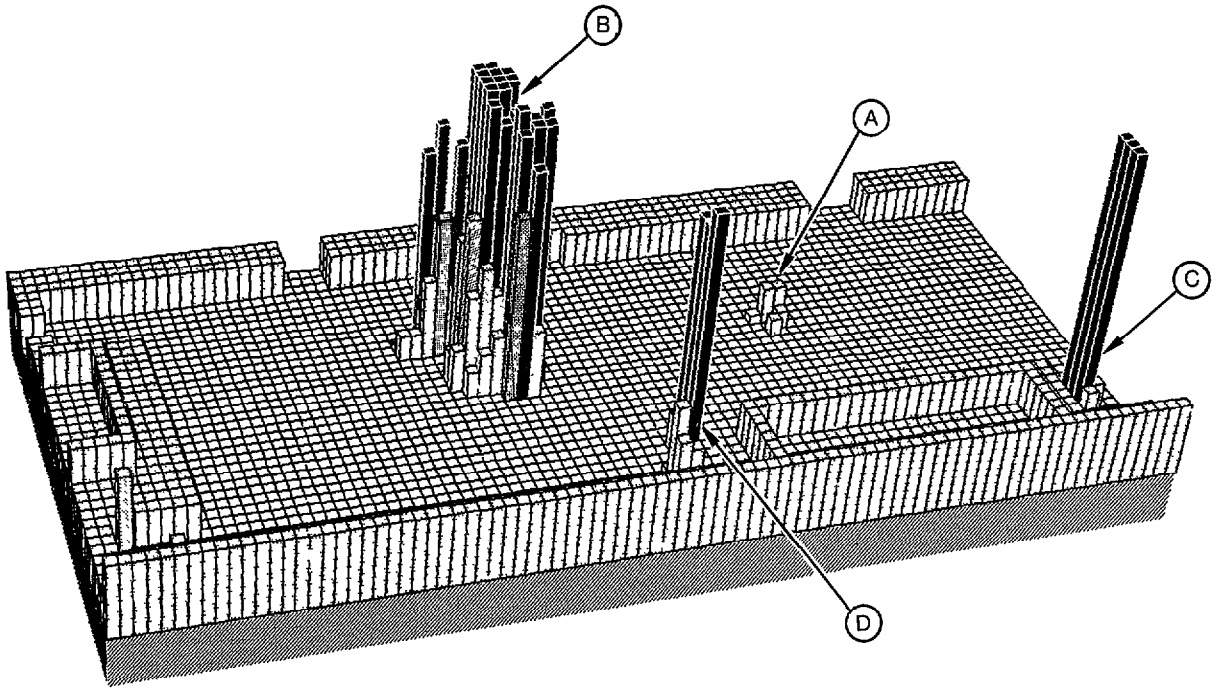


Figure 57. After third pass around the room, the old representation of chair at location A is almost completely decayed.

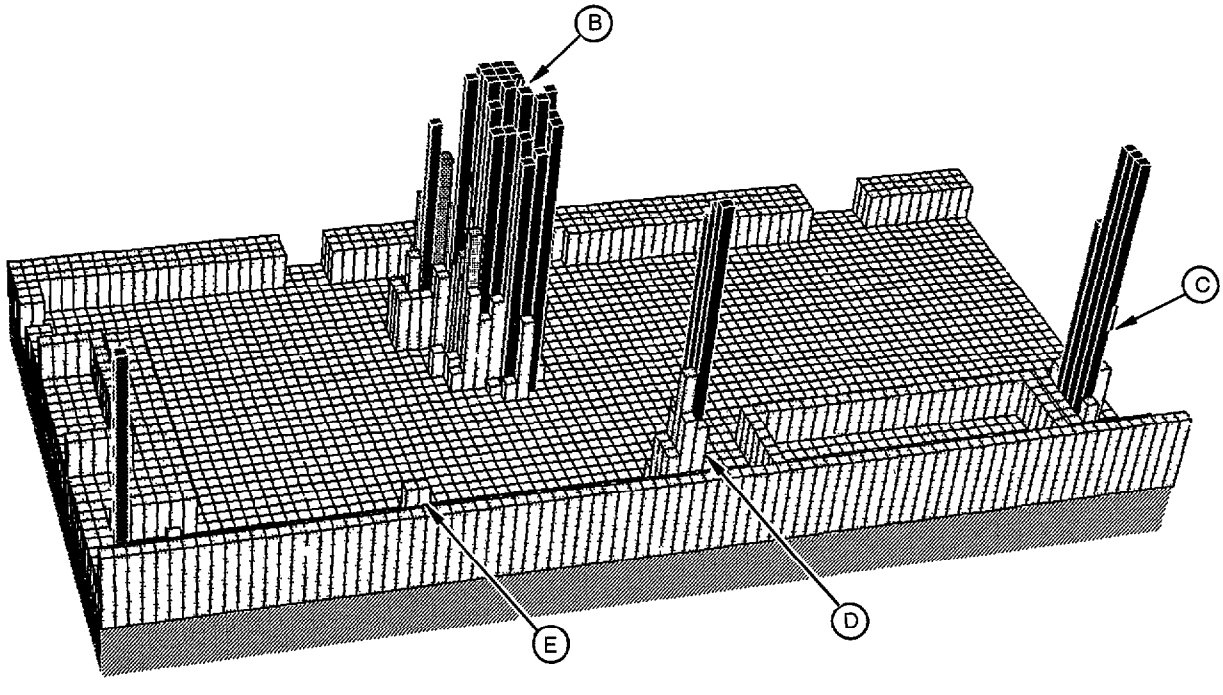


Figure 58. Final distribution after four passes; note small representation at E caused by momentary appearance of an observer in doorway.

Retrace (or backtrack) the path found by the Find_Path operation to create a list of straight-line segments describing the path from source to destination.

Create the movement commands and execute the path. If the path was successfully executed, then return successful status.

Otherwise, plan a new path to the current destination. If that fails, return to Step 1 to try to find a different path.

Each of these tasks is described in detail in the following sections.

The original path planner was based on the Lee (1961) path connection algorithm, with the cell coding enhancements suggested by Rubin (1974). The basic search algorithm begins by *expanding* the initial cell corresponding to the robot's current position in the floor map, i.e., each unoccupied neighbor cell is added to the *expansion list*. Then each cell on the expansion list is expanded. This process continues until the destination cell is placed on the expansion list, or the list becomes empty, in which case no path exists.

When a cell is placed on the expansion list, a value (arrow) indicating the direction to the parent cell is stored in the map array. Once the destination cell has been reached, retracing the path consists of merely following the directional arrows back to the source. During this process, only those points representing a change in direction (inflection points), are recorded. The entire path is completely specified by the straight line segments connecting these inflection points. The resulting path is guaranteed to be the minimum distance path.

The minimal distance path, however, is not necessarily the *best* path. Sometimes it is more desirable to minimize the number of turns, maximize the distance from obstacles, etc. The search strategy can be altered accordingly by assigning a cost to each cell prior to adding it to the expansion list; only the minimum cost cells are then expanded. This is known in the literature as an A* search (Winston, 1984).

4.2.3.1.1 Finding a Path. The Planner first makes a working copy of the appropriate floor map. As discussed, this map contains a byte for each square in the room, where the size of a square can range from one inch up to several feet, depending on the room size and desired resolution. The Find-Path routine stores two special bytes in the floor map, one indicating the current location of the robot (START), and the second indicating the desired destination (DEST). During the search process, the algorithm looks for the floor cell containing the DEST byte and similarly, during the backtrack process, looks for the START byte.

Information about the source cell (such as X-Y location, cost, distance traveled, etc.) is then put onto a *frontier* list consisting of those points on the outer edge of the search envelope that are candidates for the *expansion* process, to be discussed below. Putting the source cell on the frontier list *seeds* the algorithm so it has a cell to expand.

The algorithm then enters a loop which terminates only when there are no more cells on the frontier list, or when a path has been found. (If the frontier list is empty, then no path is possible and the search fails.) The first step inside the loop is to find all the cells on the frontier list with minimum cost and put them on the expansion list. The *cost* of a cell is typically some computation of how *expensive* it is to look at a certain cell. In a typical A* search, the cost is set equal to the distance already traveled from the starting point to the current cell, plus the straight line distance from the current cell to the destination cell. This value is guaranteed to be less than or equal to the actual total distance from the source to the destination. This particular cost function tends to make the search expand in a direction towards the goal, which usually decreases the search time.

All the cells on the expansion list are then expanded. If the destination cell is reached, a path has been found and the algorithm terminates with a solution. Otherwise, the loop continues with the new frontier list (updated by the expansion process). The expansion process looks at all the neighbors of each cell on the

expansion list; each neighbor that is unoccupied and not on either the expansion or frontier list is then placed on the frontier list. A loop is entered which terminates when all the cells on the current expansion list have been expanded. If no cells are left on the list, then a value of FALSE is returned, indicating the destination was not reached during the current expansion and further searching of the updated frontier list will be necessary.

The next cell (or first cell if this is the first time through the loop) on the expansion list is selected. A check is first made to see if this cell can be expanded. (The only cells that can be expanded are those whose corresponding byte in the floor map is zero). If the assigned value is not zero, this cell may be occupied by a temporary obstacle detected by the robot's sensors. If so, then the cell value is decremented and the cell is put back onto the frontier list to be expanded later. This technique is used to make the algorithm find a clear path (if one exists) in preference to a cluttered path. If no clear path is found, the robot may still be able to traverse the cluttered path. This technique assists the path planner in finding a path even in the presence of faulty or inaccurate sensor data.

If the current floor map cell value is zero, then the cell can be expanded; each of the cell's *neighbors* is examined to see if it is occupied or unoccupied. *Neighbor* in this case means the four cells at right-angles to the current cell, i.e., North, East, South, and West. (These are also called the *four-connected* neighbors). If the neighbor cell contains the special byte DEST, then a path has been found and the X-Y location of the cell is saved and TRUE status is returned. Otherwise, if the cell is unoccupied it is placed on the frontier list; if it is occupied, it is ignored. Finally, information used by the backtracking routine (basically an *arrow* indicating the direction to the neighbor's parent) is stored in the floor map cell corresponding to the current neighbor. Control is then returned to the top of the loop.

4.2.3.1.2 Backtracking. This process (also called retracing or segmentation) creates the list of path segments which describe the path, based on the contents of the current floor map following the Find_Path operation. The procedure is very simple. Starting with the destination cell, the following steps are performed:

The arrow in the current cell is followed to the next cell.

If the new cell contains START, then we are done.

If the direction arrow of the new cell is the same as the

current one, then the new cell becomes the current cell, and the routine returns to step 1 above.

Otherwise, the direction has changed and the current X-Y coordinate must be added to the path segment list, and the segment counter updated.

The output of the backtracking routine is a list of X-Y coordinates describing the *waypoints* through which the robot must pass in order to reach the destination.

4.2.3.1.3 Path Execution. Once a path segment list has been specified, the robot must then follow the route described to reach the desired goal. Each segment of the path is executed individually, typically consisting of turning the robot to the required heading and then performing a straight-line move of the needed distance.

Control is then passed to the segment execution routine. This procedure returns a status condition, indicating whether or not the robot was able to successfully execute the segment. If successful, then the next path segment (if any) is executed; otherwise, an error condition is returned to the caller. During segment execution the Planner performs a number of small tasks. First, it sends a command to the robot to begin moving forward the distance required by the current path segment. Next, it enters a loop looking for status packets sent back by the robot. These consist of one or more of the following:

A *move complete* report, indicating the robot has finished moving the desired distance. In this case, successful status is returned.

An obstacle report, indicating the robot has stopped because an obstacle is in its way. Obstacle status is returned.

A dead reckoning update. The current dead-reckoned position is updated.

A collision avoidance sonar packet. In this case, the sonar mapping routine is invoked and the current representation of the map is updated.

The loop is repeated until one of the return conditions is met.

4.2.3.2 Collision Avoidance. For a mobile robot to be truly autonomous, it must cope with the classic problem of avoiding an unexpected, unmapped obstacle. Initial approaches involved the development of a second localized *relative* map which represented the relative locations of objects detected in front of the robot by onboard sensors while traversing a path segment (Harrington and Klarer, 1987; Crowley, 1985; Everett, Gilbreath, and Bianchini, 1988). When range to an obstacle fell below a critical threshold, robot motion was halted and a path around the obstacle was planned, using the smaller relative map.

In this approach, however, the relative map is very transitory in nature, created at the beginning of each move and discarded at the end. The only information in the map is obtained from range sensors while the robot is in motion. However, since there is no memory of previously encountered obstacles, no learning curve exists, and several avoidance maneuvers may be required to complete the path if the area is congested.

Another possibility would be to actually encode the position of newly detected transient objects into the original *absolute* map while a path is being executed. This scheme has the advantage that all previous information about the environment is also available. Thus, in addition to avoiding newly detected objects, the Planner also knows about previously modeled obstacles which may be out of sensor range or occluded in some way. However, unlike the previous method, this technique tends to produce cluttered maps over a period of time, due to (1) the transient nature of some objects, (2) erroneous sensor readings, and (3) uncertainty in actual robot position. Eventually the workplace may appear impassable as far as the model is concerned.

A modified approach was adopted in this research which combines the best of both the relative and absolute schemes. Object space is subdivided into the two categories of permanent and transient objects. All collision avoidance sensor information is statistically represented in the absolute map, based on the number of times something was seen at a given cell location. Conversely, when a previously modeled object is no longer detected at its original position, the probability of occupancy for the associated cell is decreased; if the probability is reduced to zero, the cell is again regarded as free space.

The distinction between permanent and transient objects is an important feature which is largely responsible for the robust nature of the modeling scheme. Permanent objects remain in the model as a baseline from which to restart if the model for some reason becomes overly congested and must be flushed; only the transient objects are deleted. In addition, the Planner will always avoid permanent objects and their associated growth, whereas if necessary, the Planner will *eat through* temporary growth surrounding transient objects in an attempt to find a path. This ability was found to be necessary, as in congested environments, the growth operation often closes off feasible paths due to inaccuracies inherent in the range data. The cost of traversing transient growth increases linearly in the direction of the associated object, to minimize chances of a collision.

The use of a fixed sonar array eliminates time loss associated with the need to mechanically reposition the sensor, and the range data can be gathered and processed *on the fly*. In congested environments updates occur every 180

milliseconds, though up to 350 milliseconds may be required if longer ranges are involved. The result is a robust method of automatic map maintenance, where transient objects are added to the world map as they are encountered, and subsequently removed from the model later if no longer detected at the same location. Since previously encountered obstacles are recorded in the map, the robot can avoid them at the planning stage rather than during path execution.

Figure 59 is a 2-D depiction of a sample map created in this fashion. Free space is represented by an array value of zero and is shown in white. Permanent objects are displayed as light gray, and transient objects are displayed as black. Special cases of transient objects include doorways, which can be open or closed, and the robot's battery recharging station. The dark gray area surrounding each permanent object is the associated *growth* which allows the robot to be modeled as a point. The black areas are transient obstacles detected during the course of a move.

On completion of a path, all the transient cell probabilities are decreased by a small amount. This forced *amnesia* helps to eliminate sensor noise, and over a period of time causes all transient objects to be erased from areas seldom visited. This feature is advantageous in dynamic environments where the probability that the position of transient objects has changed is proportional to the amount of elapsed time since that object was last mapped by the robot.

4.2.4 Experimental Results

Figure 60 shows an actual map used by the path planner prior to the addition of any sonar data. A photograph (figure 61) of the same area reveals the presence of several unmapped obstacles; the positions and outlines of each have been overlaid on top of the map (figure 60) for purposes of illustration. As the Planner initially has no knowledge of these objects, the resulting path is a straight line through the area occupied by the cylinders.

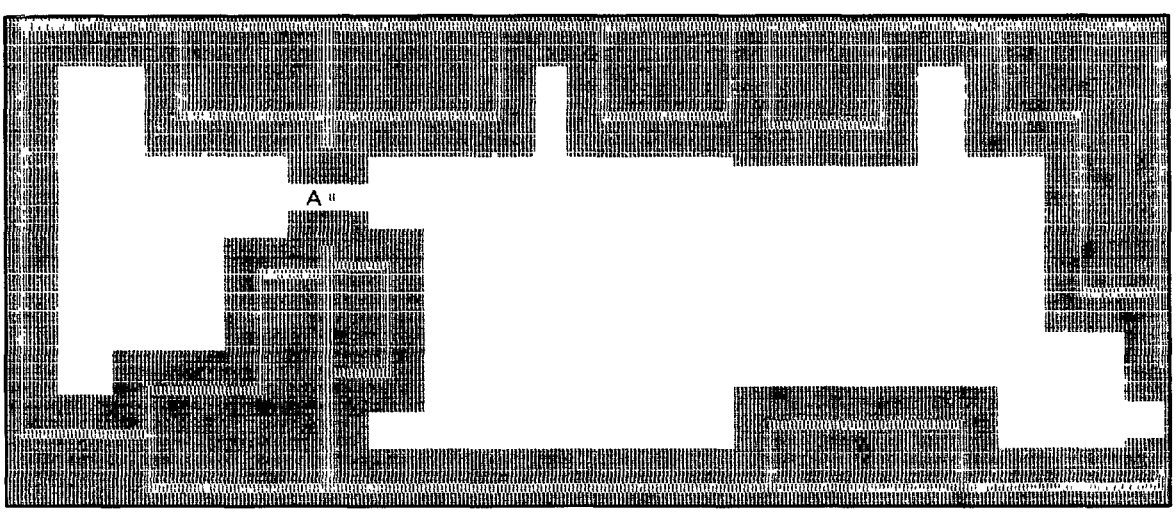


Figure 59. The darker shading around the lighter colored structure shown in this X-Y plan view of the workspace represents growth which allows the robot to be modeled as a point. The robot is located in the doorway of Point A.

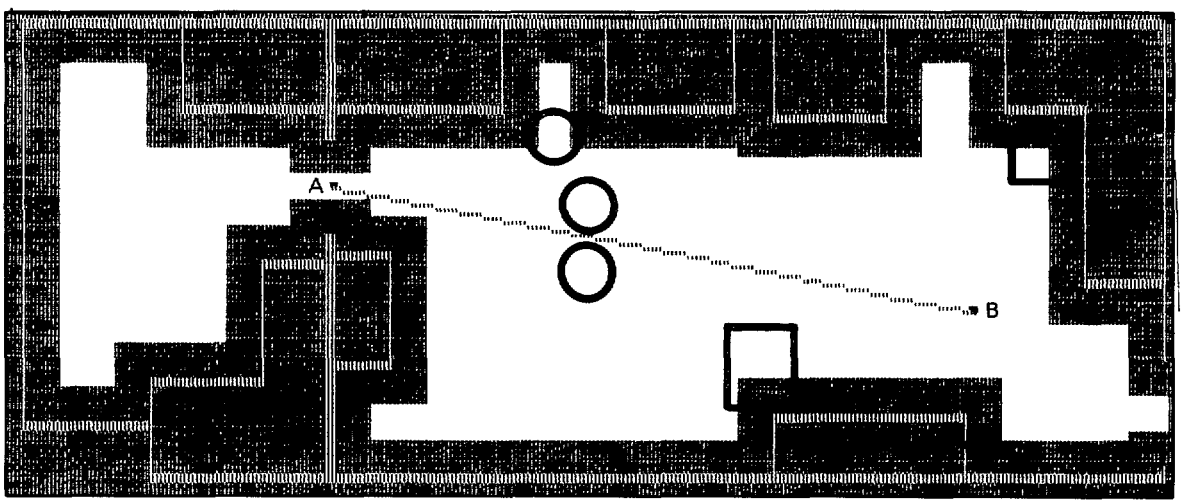


Figure 60. Results of initial find-path operation from Point A to Point B, overlaid with actual positions of three as yet undetected cylindrical objects and a rectangular cart.

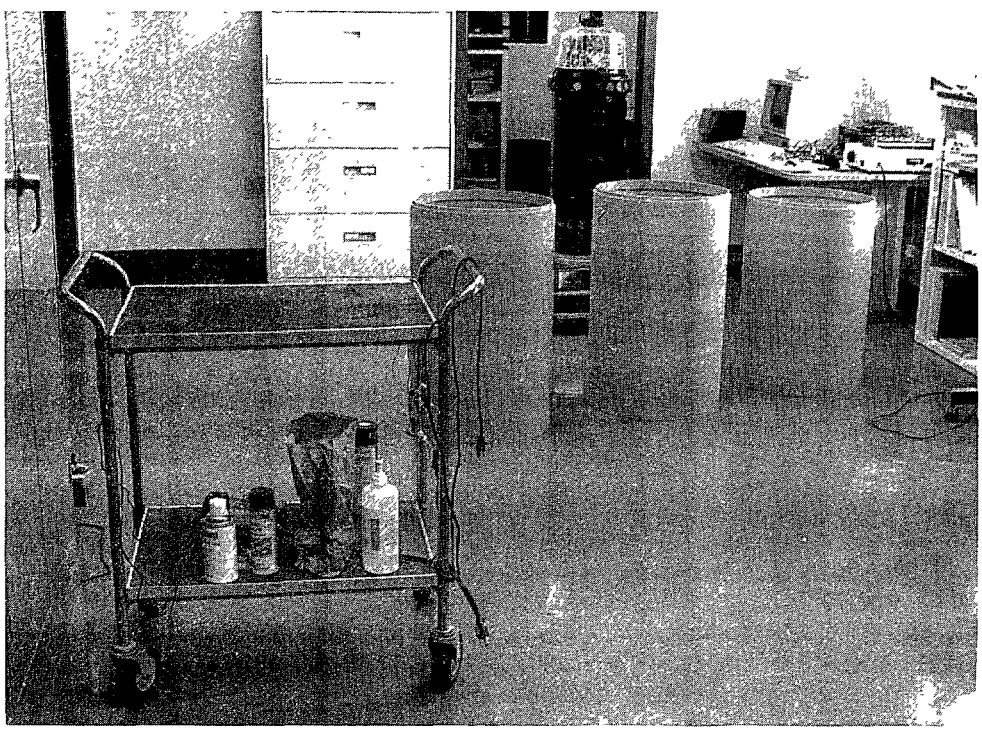


Figure 61. Photograph of the area represented in the map shown in figure 60, with the robot at the start position in the doorway, facing the cylinders.

During the execution of this path segment, the sonar array detects the row of cylinders, and begins altering the cell probabilities to reflect the perceived obstructions. When the robot has advanced to within the critical collision threshold (22 inches), forward motion is halted. At this point, all the newly mapped objects are temporarily grown (for maneuvering clearance) in preparation for the path planning operation; this transient growth is removed after the new path has been found (figure 62). Upon executing this revised path, the robot discovers the cart and plans another avoidance maneuver (figure 63).

Upon reaching its destination B (figure 64) the robot plans a path back to the original starting point A, this time avoiding the newly discovered obstacles. Since only the frontal sonars were used for mapping, the robot has not yet seen the back side of the cylinders or the cart. This necessitates the execution of another avoidance maneuver (figure 65) on the return path.

Referring again to figure 65, note the square object which has appeared in the sensor map. This perceived obstacle was erroneously added due to multipath reflection from a specular target surface. An unchecked accumulation of noise such as this would in short order render the map impassable. However, in figure 66, the object quickly disappears when there are no confirming sensor readings during the execution of the final avoidance maneuver, demonstrating how the algorithm cleans up clutter caused by anomalies in the sensor data.

4.3 HYBRID NAVIGATIONAL CONTROL

The Hybrid Navigational System seeks to combine a robust world modeling scheme (capable of planning an optimum path from the vehicle's current position to a desired destination) with a reflexive guidepath following scheme such as typically employed on AGVs. The result is a

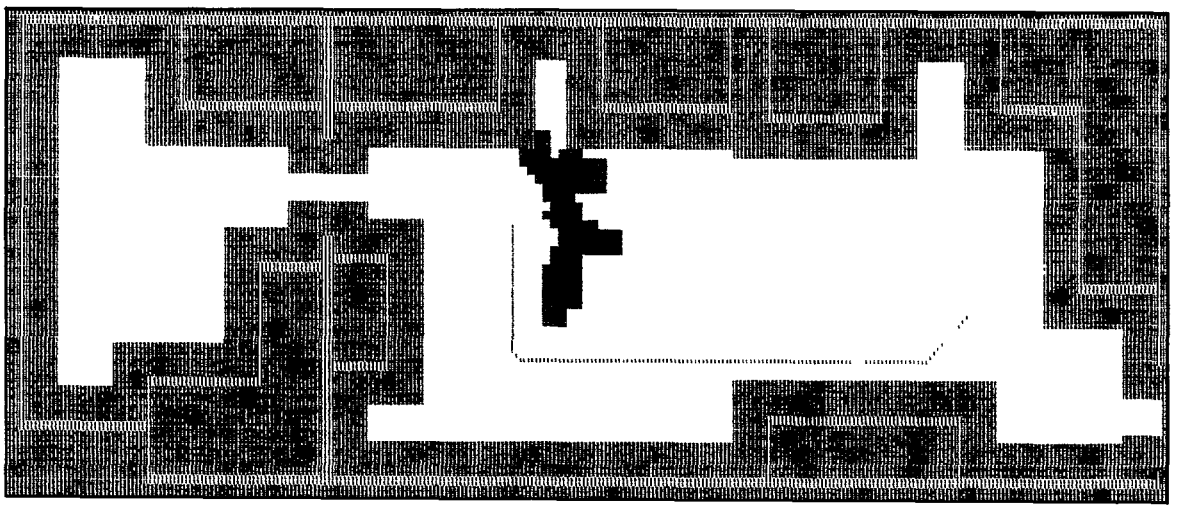


Figure 62. Avoidance maneuver generated to clear the row of cylinders shown in figures 60 and 61.

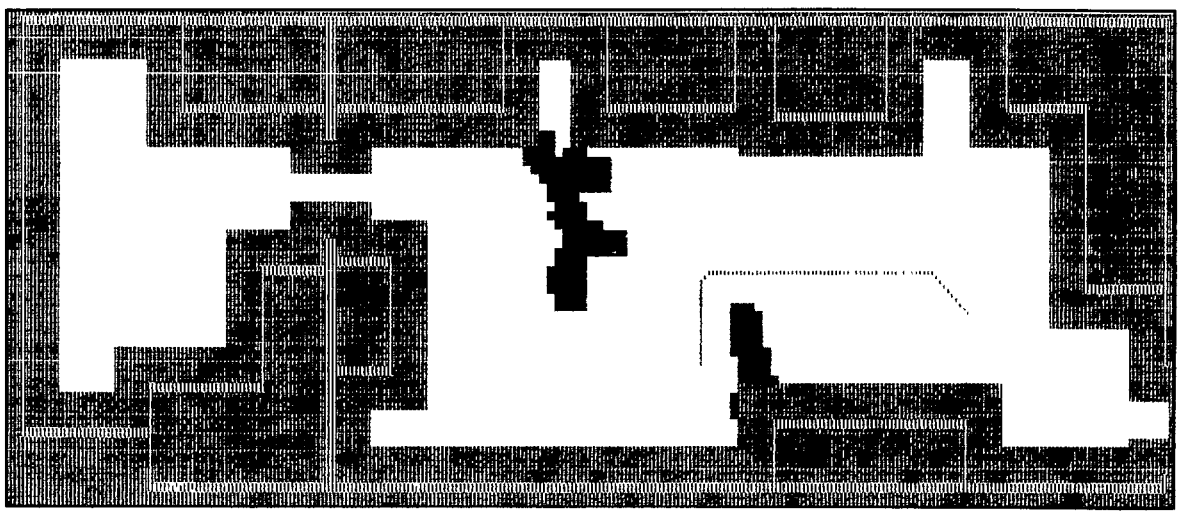


Figure 63. Revised path from robot's new position to accommodate the discovery of the cart (figure 61).

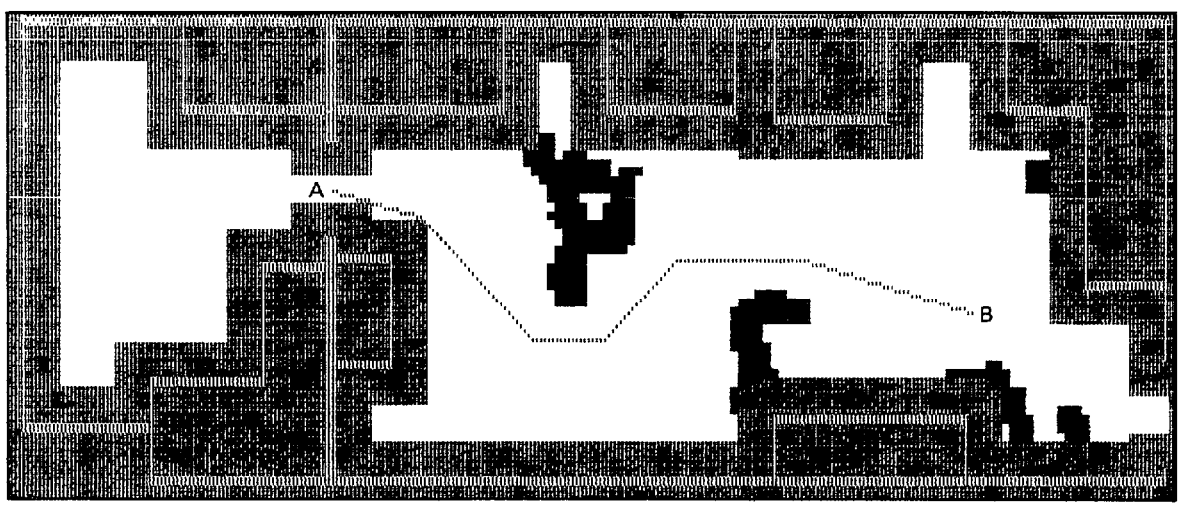


Figure 64. Return path generated by the Planner avoids the previously discovered objects.

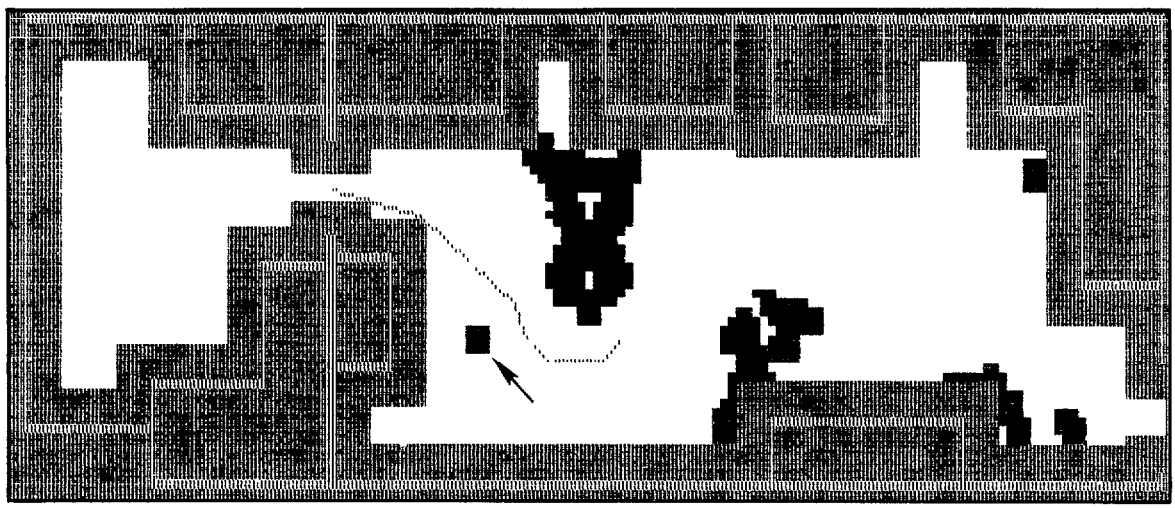


Figure 65. Erroneous *object* in lower left corner of room (arrow) resulting from multipath reflection from a specular target.

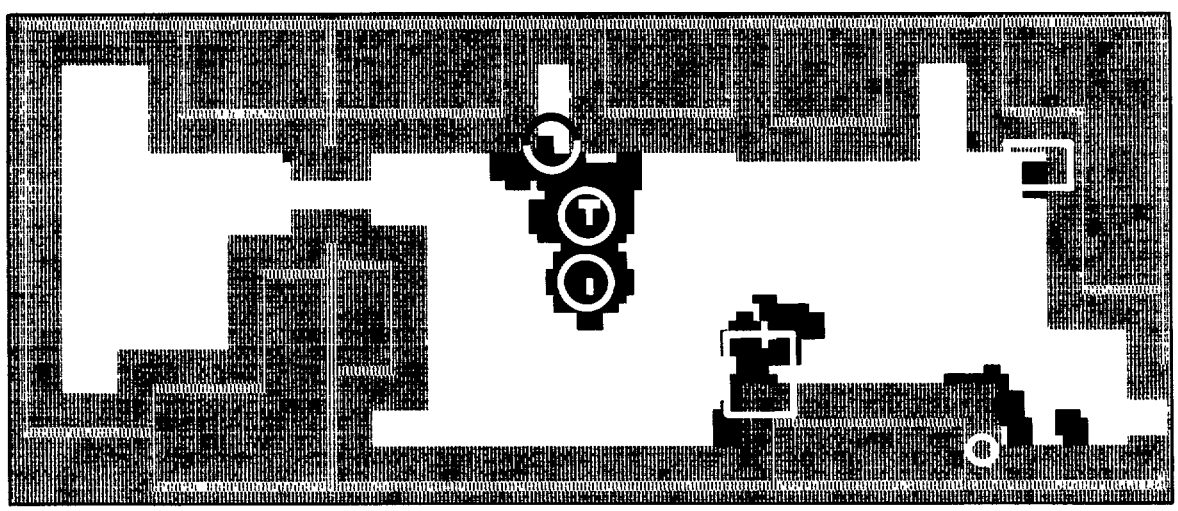


Figure 66. Final map showing removal of *noise* and refinement of object representation. The actual positions of the cylinders and cart have again been overlaid for comparison with perceived locations.

robust navigational scheme which retains the ability to dispatch the vehicle to any desired location, yet takes advantage where appropriate of designated freeways (guidepath stripes) for high speed transit and automatic positional updates. In this fashion, accumulated dead-reckoning errors that result when operating off the guidepath under absolute control are routinely eliminated upon returning to the guidepath for operation under reflexive control.

4.3.1 Navigational Reference Sensors

As stated earlier, the validity of the world model is highly dependent upon the dead-reckoning and position estimation accuracies of the robot. Minor errors in perceived heading can result in considerable positional uncertainties after straight line runs of any significant length. For this reason, several attempts were made to develop sensor subsystems that could supply updated positional and heading information at periodic intervals to the Planner.

4.3.1.1 Ultrasonic Transponder Triangulation. The first of these navigational subsystems to be investigated was an ultrasonic transponder network which consisted of a master unit onboard the robot, and three or more slaves situated around the room in known locations. The master would trigger the remote slaves via a dedicated short-range radio link (similar to that employed by the automatic recharging system), whereupon each of the slaves would emit a burst of ultrasonic energy. The slaves were each assigned specific frequencies within the ultrasonic spectrum to make them uniquely identifiable. The master onboard the robot was equipped with an ultrasonic receiver which began listening for the incoming signals, timing their respective arrivals with respect to the RF trigger signal. The software could then determine the robot's position through simple triangulation, as shown in figure 67. It was further reasoned that if the master unit was equipped with two separate receiver transducers at a known orientation and lateral separation, the robot's heading could be established as well.

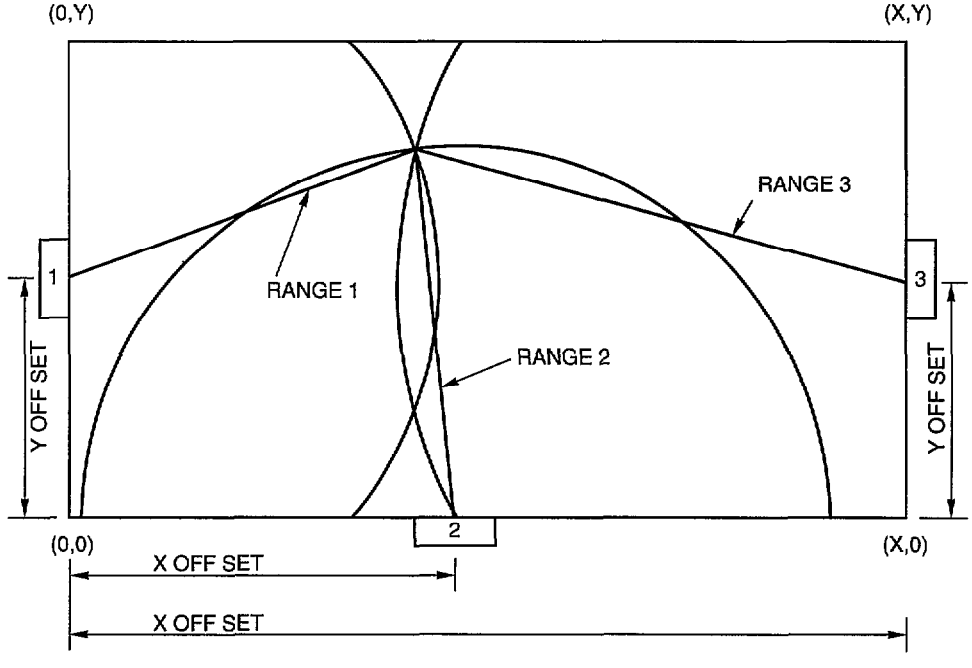


Figure 67. Example placement of three slave ultrasonic transponder units used to establish the position of the master (robot) through triangulation.

Accordingly, this concept was documented and submitted under a cooperative agreement to the Naval Postgraduate School in Monterey, CA, as a potential topic for further research, and subsequently addressed by William Dunkin, who actually built and evaluated a prototype as part of his thesis work. Although three slave units are required for an unambiguous general solution, Dunkin showed a working system could be built using just two slaves if certain conditions are taken into account (Dunkin, 1985). The resulting equations which describe the coordinates of the intersections of the two range arcs reduce to a pair of quadratics of the form

$$Ax^2 + Bx + C = 0 \qquad (29)$$

where

x = the coordinate (X or Y) being sought

$$A = 4(0F_1^2 + 0F_2^2 - 20F_1 0F_2 + D^2 \qquad (30)$$

$$B = 4(0F_2^3 - 0F_1^3 + (0F_1 - 0F_2)R_1^2 + 0F_1 0F_2^2 + (0F_2 0F_1^2 + R_2^2(0F_1 - 0F_2) - D^2(0F_1 = 0F_2)) \qquad (31)$$

$$C = F_1^4 + R_2^4 + D^4 + 0F_1^4 + 0F_2^4 + 2 - R_1^2 0F_1^2 + R_1^2 0F_2^2 + R_2^2 0F_1^2 - R_1^2 R_2^2 - R_1^2 D^2 - R_2^2 D^2 - 0F_1^2 0F_2^2 + D^2 0F_1^2 + D^2 0F_2^2 - R_2^2 0F_2^2 \qquad (32)$$

and

R_1 = measured range to slave one

R_2 = measured range to slave two

$0F_1$ = offset from reference origin to slave one, along the axis of the coordinate being sought

$0F_2$ = offset from reference origin to slave two, along the axis of the coordinate being sought

D = difference in the offsets of the two slaves along the axis of the coordinate not being sought

By solving equation 29 for all known solutions, the robot's position can be determined by comparing the set of possible solutions with the estimated position, or by using various restrictions. For example, in the setup of figure 67, the solution is constrained to only the positive quadrant of the Cartesian coordinate system shown, therefore all negative solutions are discarded (Dunkin, 1985).

Dunkin reported overall system accuracy to be ±12 inches over a test area approximately 18 by 26 feet, primarily due to poor repeatability in the propagation delays associated with the RF control link which triggered the slaves. The magnitude of this inaccuracy, obviously, precludes any attempt to derive vehicle heading from complementary solutions for a pair of receivers mounted a known distance apart on the robot, in that the ambiguity is of the same order as the maximum possible receiver separation. Significant improvement in accuracy could theoretically be obtained through optimization of the first prototype circuitry, and in fact Dunkin reported accuracies of ±3.6 inches for the same test area when the RF link propagation uncertainties were eliminated through temporary use of hardwired control lines. The advantages of a system of this type fall off rapidly, however, when the robot is to patrol in large, multiroom facilities, due to the significant complexity associated with installing multiple slaves throughout the operating area. In addition, the foregoing discussion assumes no obstructions are present to interfere with the wave propagation from the slaves to the robot, which is not always the case. A position referencing system of this type was never actually installed on ROBART for these reasons.

4.3.1.2 Fluxgate Compass. One of the first sensors actually employed on ROBART for navigational referencing purposes was a fluxgate compass manufactured by Zemco Electronics, San Ramon, CA, model number DE-700. This low-cost (around $40) unit featured a rotating analog dial, and was originally intended for

12-volt DC operation in automobiles. A system block diagram is presented in figure 68. The sensor module itself is a two-channel fluxgate magnetometer, which consists of two orthogonal sensor coils which are mutually excited by a third, driven in turn by a local oscillator at some carrier reference frequency f. The outputs V_x and V_y of amplifier channels A and B were applied across an air-core resolver to drive the display indicator. The standard resolver equations (ILC Data Device Corporation, 1982) for these two voltages are

$$V_x = K_x \sin \theta \sin (\omega t + a_x) \quad (33)$$

$$V_y = K_y \sin \theta \sin (\omega t + a_y) \quad (34)$$

where

θ is the resolver shaft angle
K_x and K_y are ideally equal transfer-function constants
a_x and a_y are ideally zero time-phase shifts

and

$\omega = 2f$, where f is the excitation frequency

Thus, for any static spatial angle θ, the equations reduce to

$$V_x = K_x \sin \theta \quad (35)$$

$$V_y = K_y \sin \theta \quad (36)$$

which can be combined to yield

$$\frac{V_x}{V_y} = \frac{\sin \theta}{\cos \theta} = \tan \quad (37)$$

Note this equation is independent of the frequency and amplitude of the reference (carrier) excitation.

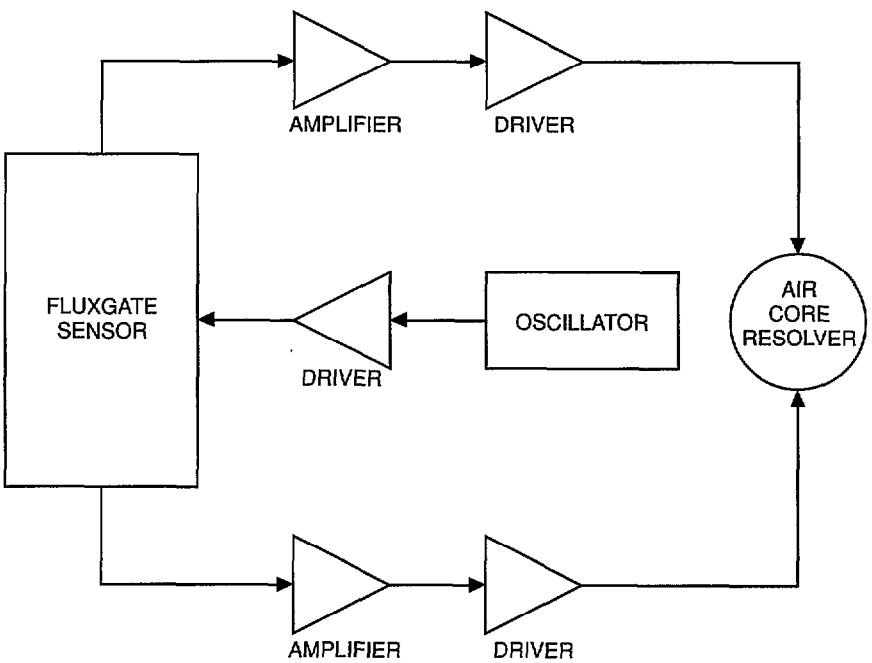


Figure 68. Block diagram of original Zemco Model DE-700 analog-dial fluxgate compass used on ROBART II.

The DC output voltages V_x and V_y thus vary as sine and cosine functions of θ, where θ is the angle of the sensor unit relative to the earth's magnetic field. The instantaneous value of θ could therefore be easily derived by performing two successive A/D conversions on these voltages, and taking the arctangent of their quotient:

$$\theta = \arctan \frac{V_x}{V_y} \qquad (38)$$

Problems associated with the use of this particular fluxgate compass, however, included a fairly high current consumption (250 milliamps), limited (8-bit) resolution of the A/D converter, and stiction in the resolver which was reflected back as load into the drive circuitry, introducing some error for minor changes in vehicle heading. In addition, the sensor itself was affected by surrounding magnetic anomalies, some of which existed onboard the robot (i.e., current flow in nearby cable runs, drive and head positioning motors), and some which existed in the surrounding environment (metal desks, bookcases, large motors, etc.).

The most serious interference turned out to be the fluctuating magnetic fields from power cables in close proximity (on the order of 12 inches) to the fluxgate sensor. As various auxiliary systems onboard the robot were turned on when needed and later deactivated to save power, the magnetic field surrounding the sensor would change accordingly. Serious errors could be introduced as well by minor changes in the position of cable runs, which occurred as a result of routine maintenance and trouble shooting. These problems were minimized by securing all cable runs with plastic tie-downs, and adopting a somewhat standardized protocol regarding which auxiliary systems would be activated when reading the compass.

There was no solution, however, for the interference effects of large metallic objects within the operating environment, and deviations of approximately 4 degrees were observed when passing within 12 inches of a large metal cabinet, for example. A final source of error was introduced by virtue of the fact that the fluxgate compass had been mounted on the robot's head, so as to be as far away as possible from the effects of the PM drive motors and numerous power distribution lines discussed above; the exact head position could only be read to within 0.82 degree due to the limited resolution of the 8-bit A/D converter. In any event, an overall system error of ±10 degrees was typical, and grossly insufficient for reliable dead reckoning calculations, but that was not the original intent of the compass. As will be discussed later in section 4.3.1.10, only a rough estimate of platform orientation (i.e., ±45 degrees) was required to initialize the beacon following system, which in turn provided the necessary navigational capability.

In FY 88 this analog compass was replaced by a newer digital version produced by Zemco, model DE-710, which cost approximately $90. The system block diagram is shown in figure 69. This unit contained a built-in ADC0834 A/D converter to read the amplified outputs of the two sensor channels, and employed its own COP 421-MLA microprocessor, which drove a liquid crystal display (LCD). All communication between the A/D converter, microprocessor, and display driver was serial in nature, with a resulting slow update rate of 0.25 Hz. The built-in LCD, however, simulated an analog dial with an extremely coarse resolution of 20 degrees between display increments, but provision was made for serial output to an external (optional) shift-register and associated three-digit numerical display. All things considered, it was determined to be more practical to discard the built-in microprocessor, A/D converter, and LCD display, and interface an external A/D converter directly to the amplifier outputs as before with the analog version. This resulted in a decrease in supply current from 168 to 94 milliamps. Power consumption turned out to be less of a factor when it was discovered the circuitry could be powered up for a reading, and then deactivated afterwards with no noticeable effect on accuracy. Overall system accuracy for this configuration was typically ±6 degrees, although a

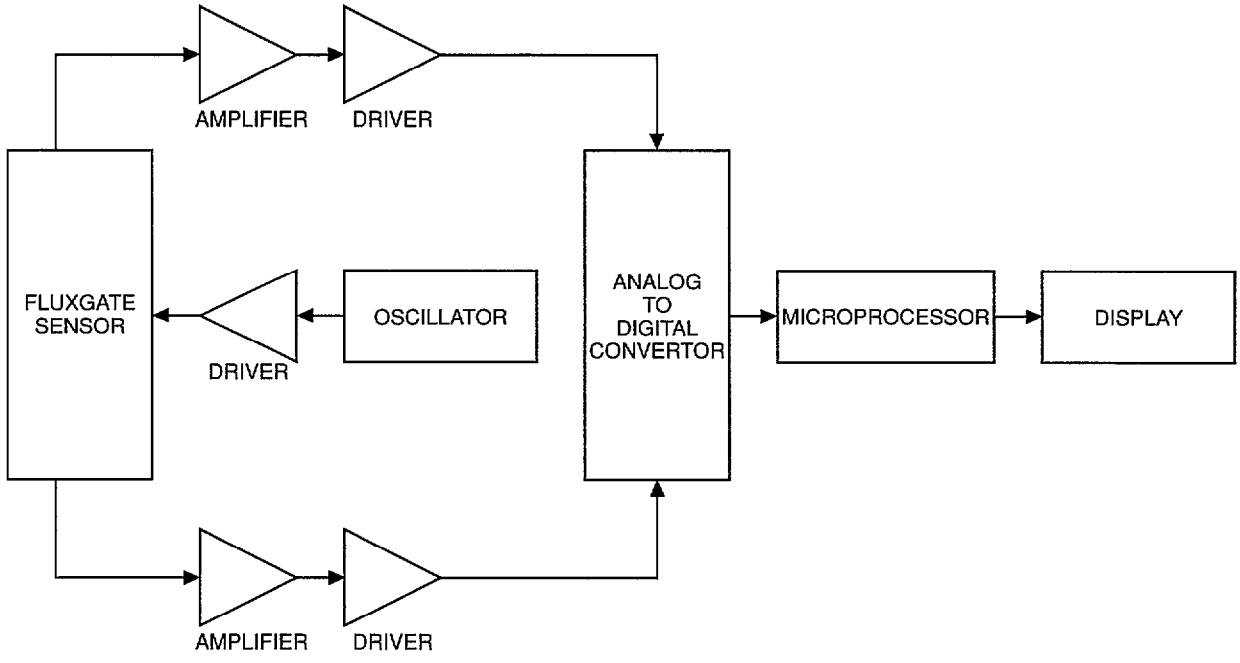


Figure 69. Partial block diagram of Zemco Model DE-710 digital fluxgate compass.

direct comparison to the analog version is not directly applicable in that the digital model was located in a slightly different place to minimize interference from nearby circuitry. The amount of effort which was put into calibration of the two systems must also be taken into account; the calibration procedure is an iterative process not easily replicated from unit to unit with any quantitative measure.

4.3.1.3 Rate Gyro. In late FY 89, a combination fluxgate compass and solid-state rate gyro package (part number FGM-G100DHS-RS232) was purchased from Watson Industries, Eau Claire, WI. The system contains its own microprocessor which is intended to integrate the information from both the rate gyro and the compass to provide a more stable output less susceptible to interference, with an update rate of 40 Hz. The system block diagram is presented in figure 70. The gyro serves to filter out the effects of magnetic anomalies in the surrounding environment, while the compass counters the long-term drift of the gyro. The system measures 2.5 by 1.75 by 3.0 inches, and weighs only 10 ounces. This is a much more expensive unit ($2500), but is advertised to have higher accuracy (±2 degrees) than the low-cost Zemco models. Power supply requirements are 12-volts DC at 200 milliamps maximum, and the unit provides a 12-bit digital output over a 2400 baud RS-232 link. The fluxgate magnetometer is gimble-mounted for improved accuracy. This system will not be installed on ROBART II, but is slated instead for the third-generation prototype, and as a result has not yet been fully evaluated.

4.3.1.4 Polarized Optical Heading Reference. Another of the initial concepts considered as a complementary heading update mechanism (providing for improved accuracy over the inexpensive fluxgate compass) called for placement of a number of modulated near-infrared sources on the ceiling above the robot's operating area. These sources would be fitted with polarizing filters of known angular orientation (i.e., referenced to true North). The modulated output of any one of these sources would automatically trigger a special head-mounted receiver whenever the robot traversed within the associated

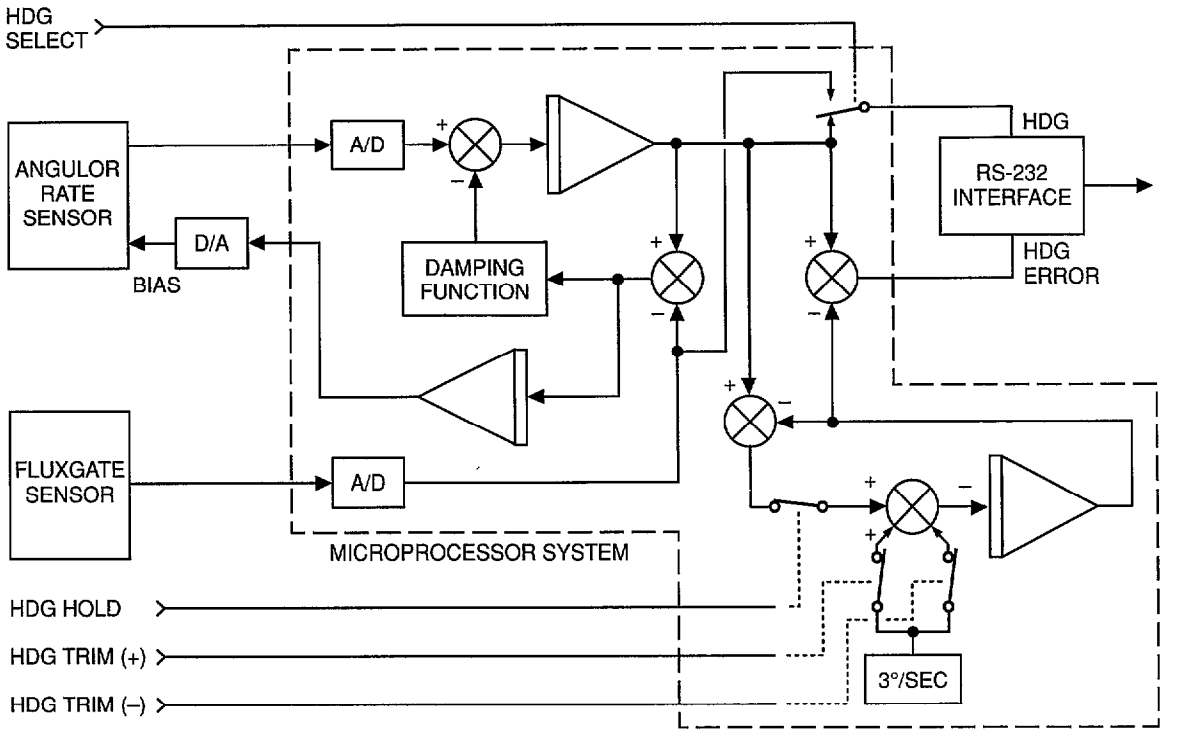


Figure 70. Block diagram of Watson combination fluxgate compass and rate gyro.

footprint of illumination of the source. An optical filter which passed only the near-infrared component (Kodak Wratten 89B) of the incoming energy would be employed to minimize the effects of ambient lighting. Upon detection of an active source, a polarizing filter mounted on the robot (just above the receiver's upward-looking PIN photodiode detector) would be rotated under servo control to ascertain the null point for which the minimum receiver output signal was observed.

This null point, of course, would be directly related to the orientation of the polarizing filter on the ceiling-mounted source, which in turn was referenced to true North as explained above. There is somewhat of an ambiguity inherent in this proposed scheme in that there would exist two null positions, 180 degrees apart, occurring when the polarizing gratings of the two filters were made orthogonal to one another. This ambiguity, however, would be resolved by choosing that null position which most closely agreed with the fluxgate compass heading. We speculated the ceiling-mounted sources could be modulated in such a way as to make them uniquely identifiable to the robot, thus allowing them to serve double duty as lateral position reference beacons. This, however, would require the PIN photodiode detector be replaced by a suitable two-axis position sensitive detector, with the appropriate electronic interface.

Although some experimentation was carried out in exploring this concept, it was never seriously pursued due to the desire to avoid modifying the robot's environment to accommodate the needs of the navigational system, and the less-than-elegant requirement to mechanically servo the angular orientation of the polarizing filter at the detector. In addition, the previously discussed head-position error would adversely affect this system as well, further complicating the integration effort.

4.3.1.5 Video Image Processing. Image processing offers an effective means of obtaining navigational information from a video

representation of the robot's surroundings. Much research has been done in this area, to include many examples of outdoor road following. For the indoor scenario of the present application, consideration was given to deriving heading information from rows of overhead fluorescent light fixtures, where available, since these are usually oriented in the direction of hallways, parallel to walls. In addition, we felt prominent discontinuities associated with doorways and windows could perhaps be readily identified and used as navigational landmarks.

The principal drawback to this approach lies in the need for relatively bulky and power hungry frame grabbing capability, memory storage, and the attendant processing requirements, which border on the impractical for small, battery-powered platforms such as ROBART. In addition, the effects of changing ambient lighting (and in some cases, the total lack of ambient lighting) must be taken into account. Accordingly, what little image processing that is addressed on ROBART is limited to line-oriented tasks, which are treated in sections 3.4.2 and 4.3.1.12 of the text.

4.3.1.6 Ultrasonic Signature Matching. This section describes one of the earliest practical techniques employed on ROBART II for periodically resetting the actual X-Y position and orientation to account for accumulated dead reckoning errors (Everett and Bianchini, 1987), based on previous work done at Carnegie Mellon University, and later by Harrington and Klarer (1986, 1987) at Sandia National Laboratories. Under this scheme, the robot must first perform a one-time *room digitizing* operation by moving in a raster-scan fashion through all unobstructed area, stopping every 12 inches to fire the upper navigational sonar array. The ring of 24 ultrasonic sensors (spaced 15 degrees apart) thus generates a database of range returns, stored in polar coordinates, with an entry for each 12-inch square floor map unit marked as free. The database entries in effect represent unique signatures of the workspace, as seen by the ring of sensors, for each X-Y position in the map. The navigational array is placed as high as possible from the ground in order to minimize distortion due to the changing positions of transient objects.

When later performing a position estimation operation, the database of the operating area is searched for a location with a signature which best matches the current set of range returns (an individual sensor return matches a database value if it falls within a specified window of acceptance, approximately 1.5 times the database resolution). Starting with the current dead-reckoned map position, the position estimator searches the database in an expanding fashion, looking for the entry (position) with the highest number of correlations matching the range values taken at the robot's present physical location. The search algorithm also skews the current position sensor data one sensor position (i.e., ±15 degrees) in each direction, in an attempt to correct for any error in current heading. If the highest number of correlations is not greater than a minimum threshold, the estimator searches a new set of neighbors farther from the original dead-reckoned position. Initial results using this technique at Sandia showed a sharp differential between the number of correspondences for a correct database match with respect to neighboring locations.

When a match is found with a high enough correspondence, the robot's position is known to within 12 inches (the database resolution), and heading to within 15 degrees. To improve the positional accuracy, the estimator will interpolate a new location within the map, using the four sensor range returns pointing 0, 90, 180, and 270 degrees relative to the robot, as long as each of these readings match their corresponding database returns within the specified tolerance.

The robot can also interpolate its heading to within about 1.5 degrees by performing several position estimations as above, rotating clockwise by 1 degree in between each estimate. As long as the computed X-Y position and heading remain the same as the previous estimate, the robot continues to rotate and take range readings. If the estimated heading suddenly changes by 15 degrees while the estimated position remains unchanged, then it is assumed the robot

has rotated approximately halfway between the previous heading and the new heading. The interpolated heading can at this point be derived by subtracting 7.5 degrees (half the rotation interval) from the most recent heading estimate. If the X-Y position changes, or the heading changes by more than 15 degrees, then heading interpolation using this technique is not possible.

This database search technique has proven to be fairly reliable for determining the robot's X-Y position during testing at NOSC, provided the operating environment does not change significantly. Some degradation is acceptable, as long as approximately 15 or more sensor readings of the 24 total are not affected. The number of correlations attained serves as a built-in indicator of database degradation, however, since as this number begins to approach the critical mark discussed above, the robot can simply initiate a new digitization routine to update the database. The only hitch here is some means of precisely monitoring the robot's position and orientation during this process is required in order to ensure the database entries are themselves valid. To date, this has meant human supervision of the room digitizing operation.

4.3.1.7 Ultrasonic Wall Referencing. Very early on in the history of the project it was perceived that stationary walls of known orientation offered an attractive potential for resetting the system heading, as well as one component of the lateral position. The concept called for positioning the robot near an unobstructed wall surface, and then sequentially firing the eleven transducers which made up the lower Collision Avoidance Sonar Array, as illustrated in figure 71a. A line-fitting operation would then be performed on the subsequent data (tables 7 and 8) from the five transducers in the vicinity of the minimum range value. (The minimum range theoretically should represent the reading from that transducer whose axis was most nearly orthogonal to the wall surface.) The angle of the resulting line with respect to the robot could then be used to adjust the robot's perceived heading based on the known orientation of the wall (figures 71b and 71c). In addition, the robot's lateral offset from this wall would be made available as well. A complete navigational update (X-Y position and heading) could thus be obtained in a corner situation at the intersection of two orthogonal walls.

Table 7. Measured sonar ranges for angular orientation of 7.5° (see figure 71b).

Range (inches)	Bearing (degrees)	X (inches)	Y (inches)
43.55	36	25.60	35.23
35.55	18	10.99	33.81
35.55	0	0.00	35.55
33.95	-18	-10.49	32.29
41.95	-36	-24.66	33.94

Linear Regression: $y = 0.0325x + 34.155$
$s^2 = 1.722$
Calculated Heading = 1.86°

Table 8. Measured sonar ranges for angular orientation of -7.5° (see figure 71c).

True heading: -7.5°

Range (inches)	Bearing (degrees)	X (inches)	Y (inches)
35.55	36	20.87	28.72
30.75	18	9.50	29.24
30.75	0	0.00	30.75
35.55	-18	-10.97	33.76
41.95	-36	-24.66	33.94

Linear Regression: $y = 0.133x + 31.143$
$s^2 = 0.754$
Calculated Heading = -7.58°

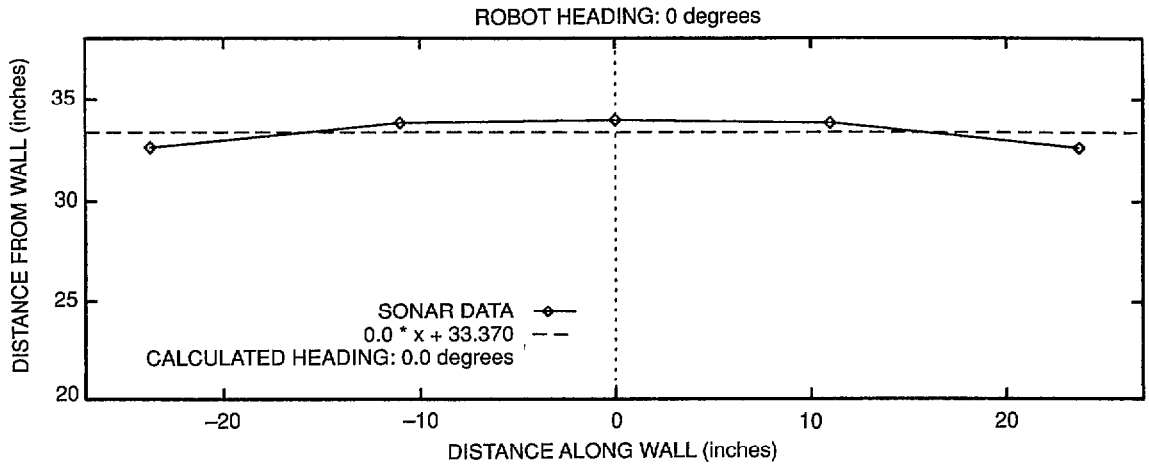


Figure 71a. A line-fit operation is performed on the range data from the Collision Avoidance Sonar Array in the vicinity of an unobstructed wall of known orientation to establish the robot's heading.

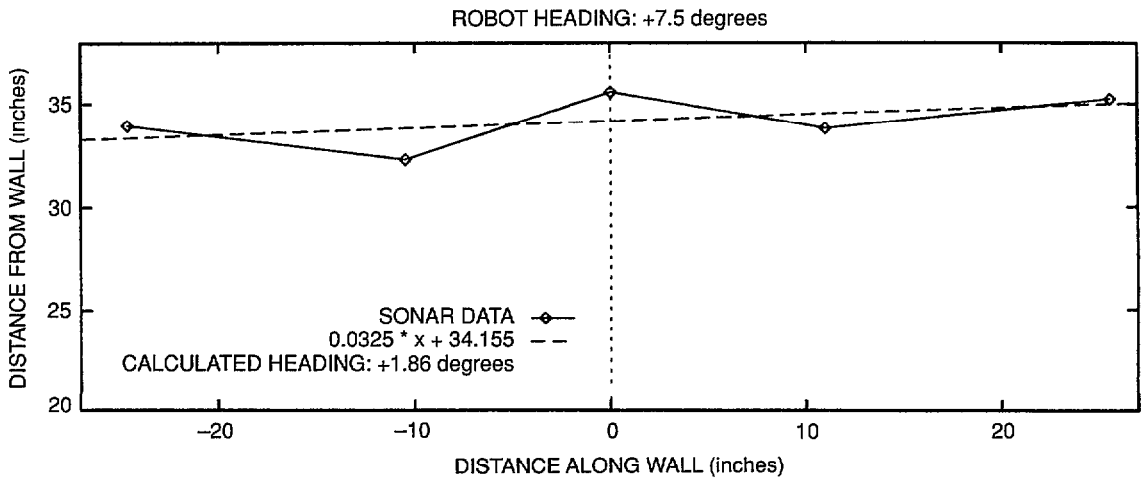


Figure 71b. A calculated angular offset of 1.86° is obtained for an actual orientation of +7.5°.

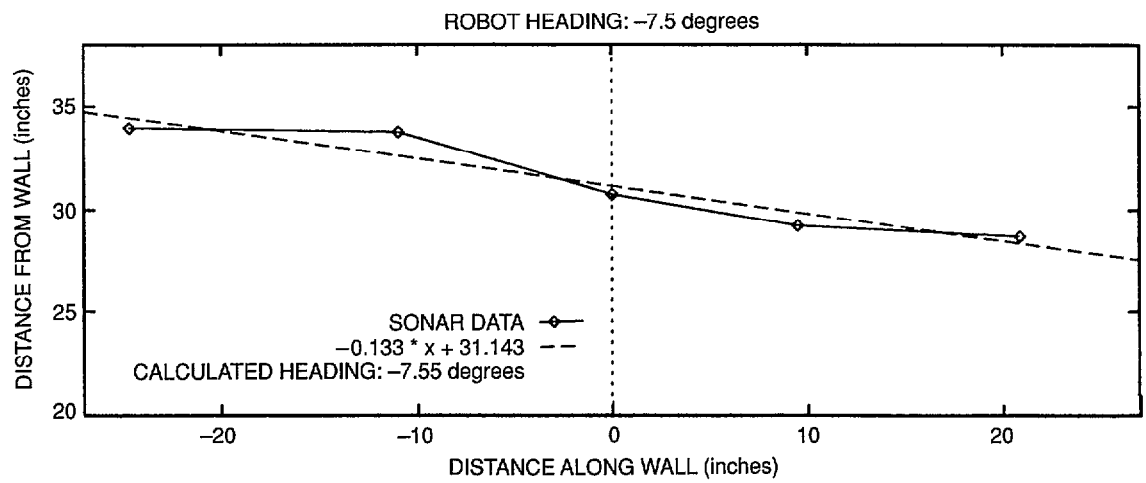


Figure 71c. A second line-fit operation is performed for an actual orientation of -7.5°, resulting in a calculated angular offset of -7.55°. Inconsistencies in performance are due primarily to specular reflection at the target surface.

Attempts to implement this concept, however, met with only limited success (figures 71b and 71c), due to problems associated with specular reflection and beam divergence, as discussed in section 3.3.2. These were aggravated by the physical orientation of the ranging sensors, which fanned out radially from the cylindrical housing. This arrangement works to your advantage when trying to detect an obstacle for collision avoidance purposes; the odds of a single transducer being nearly normal to the target surface are greatly increased. On the other hand, the odds of two or more transducers in the radial array being normal to a planar wall surface are likewise inherently low. The range values associated with those sensors which are not normal to the wall surface, obviously, are going to be adversely affected, as accuracy falls off as the angle of incidence varies from the perpendicular. Since fairly accurate data from at least three transducers is required for the wall referencing algorithm to function properly, this represents a fundamental problem.

One possible solution that was considered called for placing two or more additional ultrasonic ranging transducers along the front panel of the robot's base, which was a planar as opposed to cylindrical surface, as shown in figure 72. The robot would rotate in place to turn to the heading indicated by the axis of the minimum range value discussed above, and then fire the front panel sensors; the resulting range values should be close in value if in fact the front panel were aligned parallel to the wall. If the difference was not within a specified tolerance, the robot would rotate slightly to correct the discrepancy. Once roughly aligned in this fashion, the front panel sensors, all normal to the target surface, would provide the highly accurate range data needed by the algorithm, which would subsequently determine to robot's precise angular orientation with respect to the wall.

Alternatively, inexpensive short-range (5 to 6 feet) optical ranging systems with tightly focused beams and less susceptibility to problems associated with specular reflection could be employed for this application in place of the ultrasonic rangefinders. Unfortunately, neither of these potential solutions appeared practical due to prior design commitments in the existing prototype robot.

In an effort to minimize specular reflection with the ultrasonic sensors, however, a 2- by 4-foot sheet of molded plastic (sold as a diffuser panel for an overhead fluorescent light fixture) was attached to the wall surface as a cooperative target. The exposed surface of the plastic was made up of rows and columns of raised pyramids, approximately half an inch wide at the base, and extending outward about a quarter of an inch. This improved the overall performance somewhat, usually to within an accuracy of ±1.5 degrees, but at the expense of modifying the environment to accommodate system limitations. In addition, repeatability was somewhat lacking, and so this navigational referencing approach was ultimately abandoned. It should be noted, however, that alternative sensor selection and/or orientation, while not feasible on ROBART, could very well make this a viable method for other platforms.

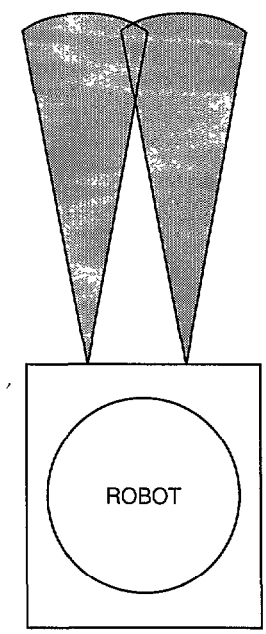


Figure 72. Potential mounting configuration for two additional sonar transducers to facilitate wall referencing.

4.3.1.8 Tactile Wall Referencing. It became apparent one obvious solution to the problems encountered with the method discussed in section 4.3.1.7 above would be to bring the robot into actual contact with the wall, properly aligned, thus eliminating any range measurement inaccuracies. The robot's heading under these conditions would be precisely 90 degrees relative to the wall orientation, and its lateral position from the wall is equally unambiguous. While not very elegant, this method is extremely robust, with the added advantage that effects of any backlash in the the drive motor reduction gears can be minimized in that both gear trains will be preloaded in the same direction.

Accordingly, this concept was quickly implemented as an interim measure pending the development of more sophisticated methods which did not require the robot to deviate from assigned functions solely for the purpose of resetting the navigational position and heading parameters.

The Planner first moves the platform to a position about 3 feet out and facing the unobstructed wall, based on the current dead reckoning information. The recalibration procedure is then requested, whereupon the Scheduler onboard the robot assumes control. With the robot stationary, the Scheduler requests a sonar update from the Collision Avoidance array, and checks to see that the robot is indeed within 4 feet of the wall. If the measured range exceeds 4 feet, an error message is sent to the Planner. Otherwise, the ranges seen by transducer #1 (mounted on the head) and transducer #8 (center of lower array) are compared; if the robot is in fact facing an unobstructed wall, these ranges should be nearly equal. If the lower range is less than the upper range by more than a specified tolerance, some obstruction is present between the robot and the wall, and this is reported to the Planner.

Assuming no discrepancies are detected, the Scheduler requests repeated updates from sonar transducer #8, and initiates forward travel, decreasing speed as the measured range to the wall falls off. When the measured range falls below 19 inches, the Scheduler checks the lower three forward-looking near-infrared proximity sensors for wall confirmation. Recall from section 4.1.2 that the sensitivities of the outer two proximity sensors are set to acquire the wall surface at a distance of 26 inches, while the center is set for a distance of 32 inches; therefore, all three should see the wall at 19 inches. If this is not the case, action is taken in accordance with the following rules:

> If none of the sensors see a target, forward motion is halted, and an error message is sent to the Planner.
>
> If the center sensor only sees a target, forward motion is halted, and an error message is sent to the Planner.
>
> If the left sensor only does not see a target, the right drive motor is halted, causing the robot to turn right.
>
> If the right sensor only does not see a target, the left drive motor is halted, causing the robot to turn left.

The software loops in this mode until all three sensors see the wall, whereupon straight-line travel is resumed, or an error condition occurs. The last two rules above have the effect of correcting any gross misalignments with the wall surface prior to impact. Note, preliminary alignment could also be accomplished in the wall approach by doing a line-fitting operation on data from the lower Collision Avoidance Array as described in the preceding section.

At this point, the robot should be moving forward towards the wall at minimum speed (1.07 in/sec). The Scheduler waits for wall impact with the tactile bumper (section 3.3.1), and stops each drive motor when the corresponding side of the bumper indicates contact. For example, if the left side of the bumper deflects first, the Scheduler stops the left drive motor, allowing the right motor to continue until such time as the right side of the bumper deflects. This turns the robot in such a way as to square it off to the wall, whereupon forward motion stops. The Scheduler then backs the platform away from the wall exactly 1 inch, and

then resumes forward travel at minimum speed for 2 seconds. Wheel slippage occurs for approximately half a second as the robot's forward travel is halted upon contact with the wall, thereby symmetrically preloading the drive reduction gears to minimize backlash errors.

With the spring-loaded tactile bumper pressed firmly against the wall, and both motors stopped, the Scheduler requests range data from the two side-looking sensors in the upper Navigational Sonar Array. One (or both) of these ranges represents the distance to a known lateral reference target, such as a bookcase (figure 73) or orthogonal wall surface. This measured range is relayed to the Planner, completing the navigational parameter update. With this information, the robot's perceived location is updated in the model, thus eliminating any accumulated dead reckoning errors. If the wall and floor surfaces were suitably equipped with contact plates for recharging the onboard batteries, this method of recalibration becomes a little more practical than would otherwise be the case, in that the robot needs to make physical contact anyway in order to recharge. Several of these stations could be situated throughout the operating area as appropriate.

4.3.1.9 Wall Following. Wall following is another type of referencing technique, similar to the method described in section 4.3.1.7, except that it takes place while the robot is traveling parallel to a wall rather than facing it. The basic procedure is described by Kadonoff (1990), and outlined below.

This technique is typically applied where the robot is traveling parallel to a wall of known position and orientation, with a specified lateral separation. During the execution of this path segment, the robot repetitively fires the ultrasonic ranging sensor which is perpendicular to and facing the wall. Over a period of time, the system will accumulate several data points, each consisting of the measured range to the wall and the associated longitudinal position of the robot along the path of travel. A straight line fit can be made to these data points, using standard linear regression techniques (Devore, 1982). If a *good fit* is obtained, (i.e., the data points do not deviate significantly from a straight line), the line is accepted, and the lateral offset from the wall as well as the current heading of the robot can be calculated as described below. With this information, the robot can adjust course to correct its heading, turning toward or away from the wall as appropriate.

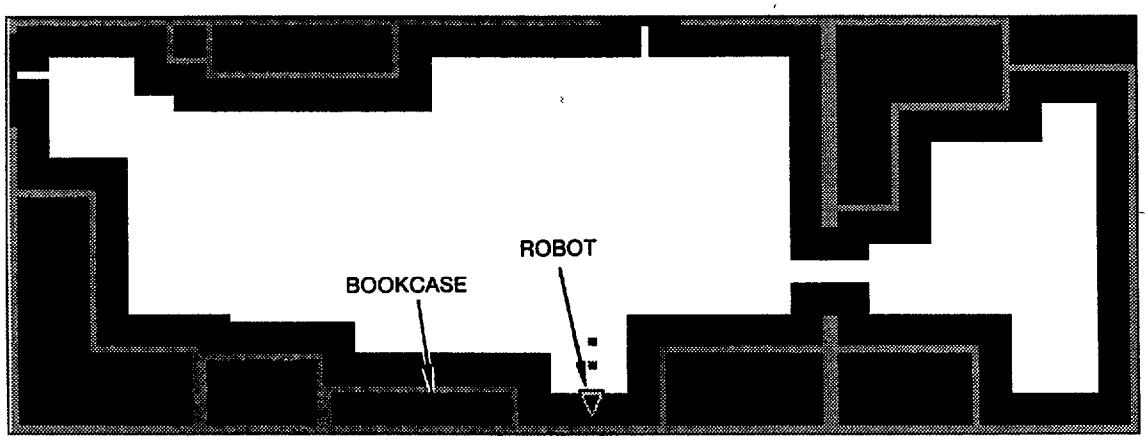


Figure 73. Sonar range to a known target such as this bookcase provides the lateral position update in the tactile wall referencing scheme.

A simple example is illustrated in figure 74. The robot begins the wall following maneuver at point A and proceeds to point B, with the measured sonar ranges indicated in the figure by lines emanating from the robot and terminating somewhere near the wall shown at the bottom. Table 9 lists the range data collected by the side-looking sonar as a function of longitudinal displacement along path segment AB.

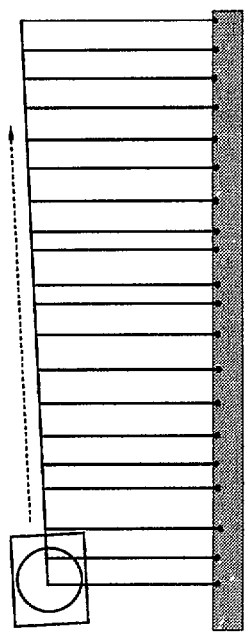


Figure 74. A line-fit operation is performed on several sonar range readings taken while the robot is in motion when wall-following.

The linear regression equations used to calculate the slope, intercept and estimated variance are as follows:

$$\text{slope} = \frac{n * \sum x_i * y_i - (\sum x_i) * (\sum y_i)}{n * \sum x_i^2 - (\sum x_i)^2} \quad (39)$$

$$\text{intercept} = \frac{\sum y_i - (\text{slope}) * (\sum x_i)}{n} \quad (40)$$

$$s^2 = \frac{\sum y_i^2 - (\text{intercept}) * \sum y_i - (\text{slope}) * \sum x_i * y_i}{n-2} \quad (41)$$

Table 9. Measured versus actual range readings along path segment AB of figure 74 (inches).

Longitudinal Position	Measured Sonar Range	Actual Reference Range
0.0	33.2	33.7
5.4	33.7	33.7
11.2	33.8	34.2
19.3	34.2	34.7
23.9	34.6	34.7
29.4	34.8	35.2
36.0	35.3	35.6
42.7	35.5	36.1
49.9	35.9	36.1
55.9	36.1	36.6
59.8	36.2	36.6
66.8	36.4	36.6
72.2	36.5	37.1
78.4	36.6	37.1
84.8	36.8	37.4
90.4	37.2	37.4
96.6	37.6	37.9
102.5	37.9	37.9
108.0	38.0	38.4
114.0	38.2	38.4

where

n = number of sonar readings taken (20 in this example)

Using these formulas, the equation of the line resulting from the use of the sonar range values is

$$y = 0.0416\,x + 33.885 \qquad s^2 = 0.0530 \quad (42)$$

while the equation of the line using the robot's measured position from the wall is

$$y = 0.0420\,x + 33.517 \qquad s^2 = 0.0335 \quad (43)$$

Figure 75a shows the measured sonar data overlaid on top of the line corresponding to equation (42), while figure 75b shows the actual reference data overlaid on top of equation (43). Figure 75c shows a comparison of the two lines: the slopes are extremely close, and the sonar data is offset from the reference data by only 0.03 foot (0.36 inch).

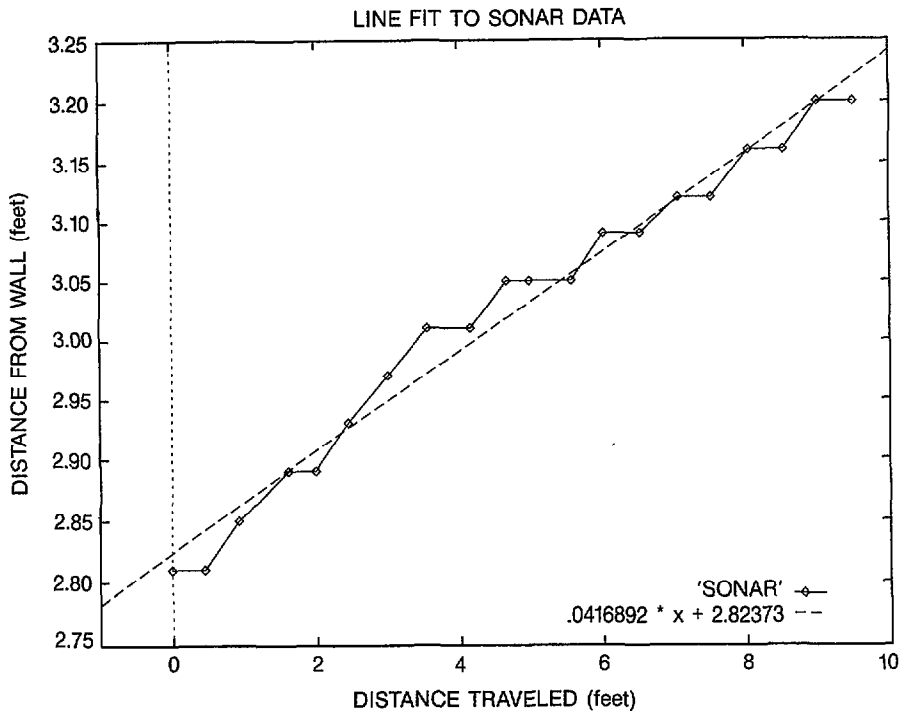


Figure 75a. Plot of measured sonar data and resulting least-squares fit.

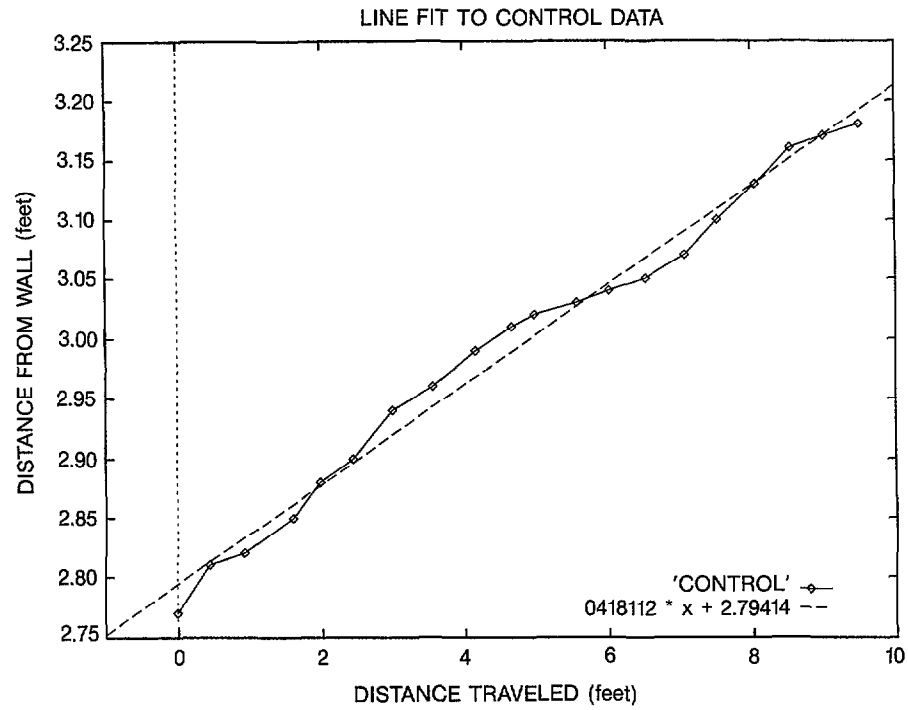


Figure 75b. Plot of actual (reference) data and associated least-squares fit. Undulations in the data are caused by imperfections in the wall itself.

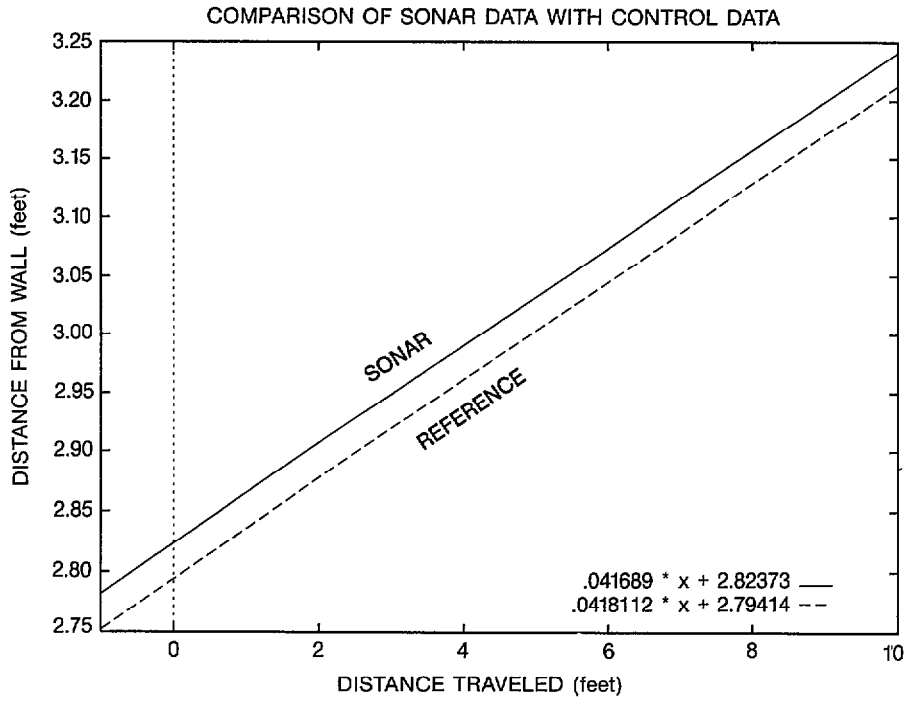


Figure 75c. A comparison of the sonar and reference line-fits of figures 75a and 75b. Note the lines are nearly parallel, with a lateral offset of less than 0.03 foot (about a third of an inch).

The robot's heading with respect to the wall can be calculated by taking the arctangent of the slope. For the sonar data, this would be

$$\theta = \tan^{-1}\ (0.0416) = 2.382° \qquad (44)$$

while for the measured (reference) data

$$\theta = \tan^{-1}\ (0.0420) = 2.405° \qquad (45)$$

In this particular example, the estimated heading of 2.382 degrees is in error by only 0.023 degrees.

While wall following will effectively reset the robot's heading and either the X or Y coordinate as discussed above, the other position coordinate remains unknown. Another technique, (developed by John Holland of Cybermotion) known as *wall approach*, can sometimes be used in this situation to reset the remaining coordinate. The robot is known to be approaching a wall directly ahead, and has been told that at the end of the path segment it should be a certain distance from this wall. Knowing the absolute position of the wall, the robot can then update the unknown coordinate using the measured range in the forward direction upon completion of the move.

4.3.1.10 Beacon Following. The hallway navigation scheme employed on ROBART I was based in part on the concept of beacon following; the recharging station was equipped with a near-infrared homing beacon, and suitably positioned in a known location to assist the robot in

entering the hallway. Once in the hallway, the robot would move parallel to the walls in a reflexive fashion, guided by numerous near-infrared proximity sensors. The robot could determine its general orientation in the hallway if told beforehand which direction afforded a view of the beacon. With additional prior knowledge of where the rooms were situated with respect to this hallway, the robot could proceed in a semi-intelligent fashion to any given room, simply by counting off the correct number of open doorways on the appropriate side of the hall. A more simplistic precursor to the programmable near-infrared ranging system presented in section 3.3.4 was used to locate the doorway openings (Everett, 1982).

Other early work incorporating beacon tracking was performed at the Laboratoire d'Automatique et d'Analyse des Systemes (Toulouse, France), and involved the development of a navigation subsystem for determining position and orientation of the mobile robot Hilare (Banzil et al., 1981). The system consisted of two near-infrared emitter-detectors mounted with a 25-cm vertical separation on a rotating mast, used in conjunction with reflective beacon arrays at known locations in three corners of the room. Each of these beacon arrays was constructed of retroreflective tape applied to three vertical cylinders, spaced in a recognizable configuration as shown in figure 76. One of the beacon arrays was inverted as shown in order to be uniquely distinguishable for purposes of establishing an origin. The cylinders were vertically spaced so as to intersect the two planes generated by the rotating optical axes of the two stacked emitter-receivers on the robot's mast. A detected reflection pattern such as shown in figure 77 confirmed beacon acquisition. Angular orientation relative to each of the retroreflective arrays was inferred from the stepper motor commands which drove the scanning mechanism; lateral position was subsequently determined through simple triangulation.

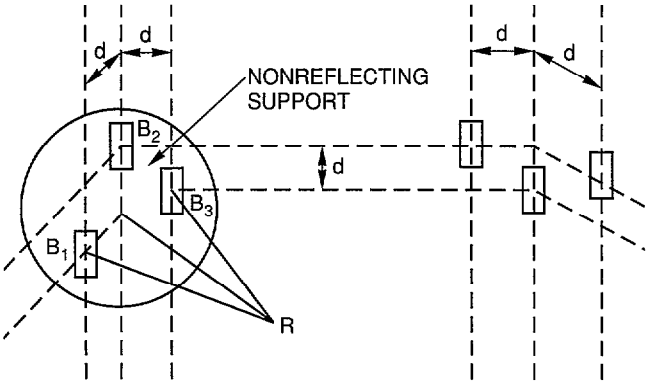


Figure 76. Beacon array configuration used by the navigational scheme employed on the mobile robot Hilare.

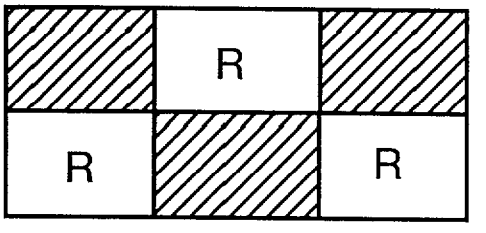


Figure 77. Valid reflection pattern resulting from the array configuration of figure 76.

NAMCO Controls (Mentor, OH) developed a multifunction laser-based sensor system known as LASERNET, intended primarily for AGV applications in industrial environments. This active scanning device requires retroreflective targets in order to measure range and angular position. A servo-controlled rotating mirror horizontally pans a helium-neon laser beam through an arc of 90 degrees (45 degrees either side of the center) at a 20-Hz update rate. When the laser beam sweeps across a retro-reflective target

of known dimensions, a return signal of finite duration is sensed by the detector. Since the targets are all the same size, the return generated by a close target will be of longer duration than from a distant one (figure 78). In effect, the target appears larger.

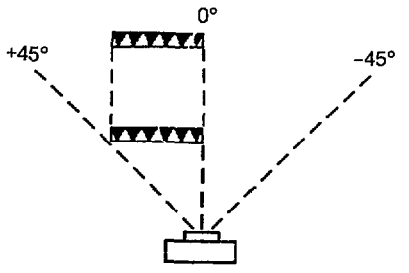


Figure 78. Energy emitted by a scanning laser is reflected from a retroreflective target of known size to calculate target range in the NAMCO LASERNET scheme.

Range is calculated from the equation

$$d = \frac{W}{2\tan(VT_a/2)} \qquad (46)$$

Where: d = range to target
W = target width
V = scan velocity (7200 deg/sec)
T_a = duration of the returned pulse

Because the target width and angular scan velocity are known, the equation reduces to an inverse function of the pulse duration, T_a. With 4-inch targets, the effective range of the sensor is from 1 to 20 feet, with an accuracy of ±4 percent at 20 feet, and a range resolution of 9.6 inches at 20 feet down to 0.1 inch at 1 foot.

Angle measurement is initiated when the scanner begins its sweep; the laser strikes an internal synchronization photodetector which starts a timing sequence. The beam is then panned across the scene until reflected by a retroreflective target in the field-of-view. The resulting returned signal is detected by the sensor, terminating the timing sequence (figure 79). The elapsed time is used to calculate the angular position of the target in the equation

$$\theta = (VT_b) - 45 \qquad (47)$$

Where: θ = target angle
V = scan velocity (7200 deg/sec)
T_b = interval between scan initiation and target detection

This angle calculation determines the position of the leading edge of the target with respect to the mid-point of the 90-degree scan.

Note all of the above calculations assume the target is positioned perpendicular to the angle of incidence of the laser source. If a target happens to be rotated or otherwise skewed away from the perpendicular, it will appear narrower, with a resultant range measurement error. Errors in angle determination also occur because the leading edge is either positioned in front of or behind the center of the target. With the proper placement of retroreflective targets or tape, however, the Lasernet system can guide AGVs using wall following, beacon following, or track following methods. Because of the requirement for fixed-position retroreflectors, the use of such a system in mobile applications will in general be limited to facility robots which work in known semistructured environments.

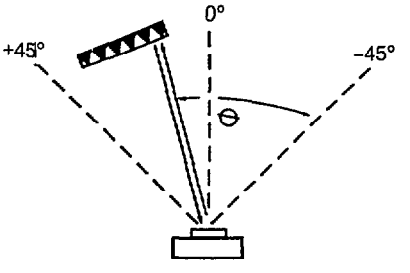


Figure 79. The elapsed time between sweep initiation and leading-edge detection can be directly converted into target bearing.

One of the principal problems associated with beacon following systems, aside from lack

of flexibility and the obvious requirement to modify the environment, arises from the need to preserve a clear line-of-sight between the robot and the beacon. This unobstructed view is sometimes difficult if not impossible in certain applications, such as warehouse environments, where space is a premium. For this reason, Rathbone, Valley, and Kindlmann (1986) describe a proposed system for AGV guidance that employs an upward-looking imaging sensor, able to positively identify and track a number of specially coded near-infrared LED beacons mounted on the ceiling. Sufficient redundancy is provided to account for the fact that some beacons may be occluded from view from time to time.

4.3.1.11 Doorway Transit Referencing. The concept of using existing interior doorways as navigational landmarks has always been appealing, in that no modifications to the surrounding environment are required. The robot by necessity must travel through a doorway to enter an adjoining space; if in so doing the system could obtain an accurate positional update, then such would indeed represent an elegant solution to the problem of cumulative dead reckoning errors. One approach to implementing this concept can be decomposed into the following tasks:

Finding the doorway.

Entering the doorway.

Verifying the doorway.

Determining longitudinal position relative to doorway.

Determining lateral position relative to doorway.

Determining heading (angular orientation) relative to doorway.

The first of these tasks is addressed through use of a combination of ultrasonic ranging sensors, which have good distance measurement capability but poor angular resolution, and optical proximity sensors, which typically have superior angular resolution, but little or no ranging capability. In addition, the problem is greatly simplified by virtue of the fact the Planner knows where the door is located within the map structure, and can direct the robot reasonably well to the vicinity of this position. In addition, the Planner always orients the path segment that actually penetrates the door opening to be orthogonal to the associated wall. With such *a priori* information, the task of finding the doorway's actual position with respect to the robot is greatly simplified.

To accomplish this task, the Planner informs the Scheduler that the current path segment penetrates a door opening, and provides the estimated bearing and distance to the door. The Scheduler rotates the head to this bearing (typically straight ahead), thus pointing the long range near-infrared proximity sensor (section 3.3.4) at the center of the passage. Unless the robot is significantly misaligned due to accumulated dead reckoning errors, the proximity sensor will return a *no target* condition, as it should be looking through the open doorway. If this is not the case, the head begins scanning 15 degrees either side of centerline in an attempt to find the opening. If this search fails to locate the doorway, an error condition is returned informing the Planner that the robot is either significantly lost to where the door penetration routine won't work, or the door is closed.

Assuming the opening is detected, the Scheduler next attempts to locate the left and right edges by panning the head and watching the proximity sensor output for a *target* condition, indicative of energy being reflected from the door casings (see doorway detail, figure 80) and adjacent wall areas to either side. Head position angles corresponding to the left and right boundaries are then averaged to yield a relative bearing to the actual center of the doorway.

The Scheduler alters the robot's heading to be coincident with this bearing, and begins looking at the sonar data from the center five transducers in the Collision Avoidance Array for range confirmation. Measured distance to the door should be within a specified tolerance of the estimated range provided earlier by the Planner, less distance traveled in the interim, otherwise another error condition is returned. If the robot is more than 5 feet from the doorway,

the center three transducers should all indicate ranges within this window of acceptance. As the robot closes on the doorway, the beam from the center transducer should eventually break through the opening, with a corresponding increase in range to target. This occurs at the point where the effective beam width at the indicated distance becomes less than the width of the doorway, assuming the robot is perfectly aligned with the center of the opening. (Perfect alignment is typically not the case, however, resulting in a slight delay as the beam narrows further on approach, before the jump in range is observed.)

At the instant the center beam penetrates the opening, the two adjoining beams from transducers 7 and 9 should by virtue of their orientation in the array be directed at the left and right door casings, as shown in figure 80. The respective range readings from these two

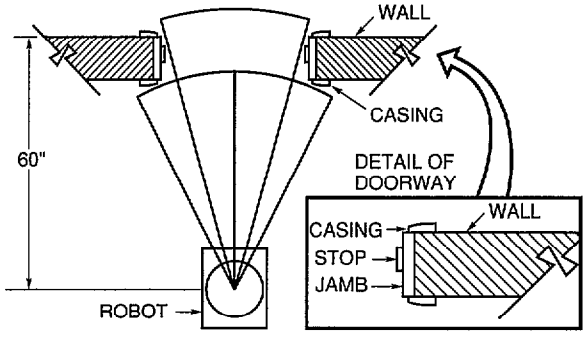


Figure 80. Energy is reflected from the left and right door casings, but the center beam penetrates the opening at a distance of about 5 feet.

transducers at this point should again be consistent with the previously estimated range to the doorway, until such time as the indicated ranges decrease to around 36 inches, whereupon these beams should break through the opening, as shown in figure 81. If either of these ranges decreases below 12 inches prior to penetration, the robot is likely to impact the side of the door, and the Scheduler will have to execute a corrective maneuver to attain better alignment.

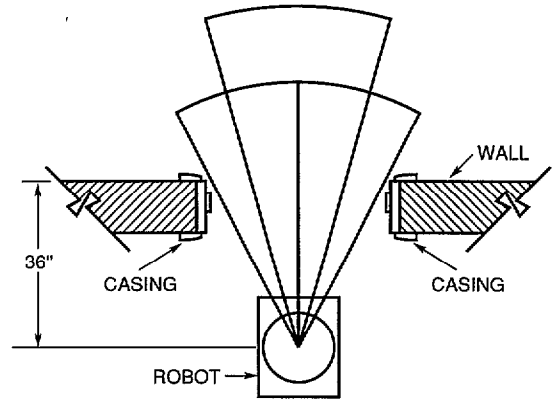


Figure 81. As the robot closes on a 36-inch doorway, all three sonar beams should penetrate the opening at approximately 36 inches.

It may appear the robot's alignment with the doorway could be calculated in advance at a distance of approximately 4 feet by simply comparing the range returns from transducers 7 and 9, but this method turns out to be unreliable due to the possibility of furniture or other objects on either side of the door, which will intercept the beam prior to its striking the door casing. In addition, doorways are sometimes placed in the corner of a room in close proximity to an adjoining wall, which will interfere with the sonar readings on that particular side. For this reason, it was found necessary to let the robot get very close to the opening as discussed above before assessing alignment.

The next step in the procedure calls for deriving X-Y positional data while passing through the door opening. The most obvious solution for the transverse fix is to ping the two side-looking transducers in the upper Navigational Sonar Array at the left and right door jambs; the only difficulty here would be knowing exactly when to ping. One solution might be to ping continuously during transit, and then use the minimum range value thus obtained. A second approach would be to estimate the distance to the center of the opening from the last set of ranges measured by transducers 7 and 9 just prior to penetration, and then ping the door jambs after traversing that amount of distance. In either case, the left and right range readings

thus obtained specify the robot's lateral position, and for purposes of verification should add together to yield the width of the door passage, typically 36 inches.

The task of obtaining a longitudinal fix during doorway transit is a little more difficult. The longitudinal fix could be derived from the last set of readings obtained by transducers 7 and 9 mentioned above, but the accuracy would be somewhat suspect. Alternatively, if the transverse fix discussed above is obtained by successive pinging of the door jambs, then post analysis of the data should yield a door edge profile in the sense that ranges to either side will decrease to some minimum upon entry, remain at that minimum plus or minus some tolerance value for a finite length of time which is proportional to the width of the jamb (thickness of the wall), and then increase. The midpoint of this period of minimum ranges would then correspond to the midpoint of the door jamb width (centerline of the wall), which is of course the desired longitudinal fix.

Both of the above solutions, however, assume an ideal door opening in the center of an unobstructed wall, and will suffer significantly from the presence of objects near the open doorway, not the least of which might be the door itself. (When in the open position, the door folds back to one side, adding several inches in projected target surface which will interfere with the ranging process.) This is primarily due to problems associated with specular reflection and beam divergence in the ultrasonic rangefinders employed.

Diffuse-mode near-infrared proximity sensors are often employed in an effort to compensate for some of the limitations in ultrasonic systems, in that the beams can be tightly focused, and specular reflection is less significant due to the shorter wavelengths involved (Everett, 1988a; Banner Engineering Corporation, 1989). This type of proximity sensor provides no range measurement capability, however, other than that which can be inferred from the strength of returning energy, which varies as a function of target reflectivity. If the sensors are horizontally mounted on the robot so as to be orthogonal to the direction of travel, however, they could be used to detect the leading edge of the door casing as the robot passed through the opening. As shown in figure 82, the elapsed time between target detection by sensors mounted on either side of the robot could be used to calculate as well the angular orientation of the robot with respect to the doorway, in accordance with the following formula:

$$\sin \theta = \frac{vT}{d} \qquad (48)$$

Where: θ = angular orientation
v = velocity of robot
T = elapsed time between detections
d = target separation distance

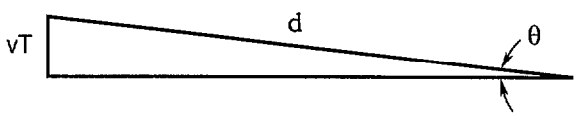


Figure 82. Elapsed time between door frame detection by left and right proximity sensors with known separation can be used to calculate the robot's heading.

To achieve any useful accuracy in deriving the heading of the robot in this fashion, however, the following conditions must apply:

The sensors must be rigidly mounted to retain their orthogonal relationship to the robot.

The sensors must have well defined narrow beams.

The excess gain must be sufficiently high to ensure rapid detection as the targets move into view.

The time between left and right target detection must be accurately measured.

The robot's heading must remain constant for this period.

The distance traveled by the robot during this period must be accurately measured.

The targets must stand out clearly from their respective backgrounds with no ambiguity.

The lateral separation d between targets must be known in advance or measurable during transit.

The first six conditions outlined above are easily met, but the latter two pose a problem. As previously discussed, objects on either side of the doorway can effectively mask the location of the door casing to make accurate leading-edge detection impossible. One way around this would be to apply strips of retroreflective tape to the door casings to create cooperative targets, and reduce the gain of the proximity sensors to where only these strips triggered a detection.

The use of retroreflective tape, however, requires the environment be modified to accommodate the robot, which is not in keeping with the objective of using existing (unmodified) doorways as navigational aids. Such strips are somewhat obtrusive and distracting to humans, and can be accidentally removed or painted over by maintenance crews. In addition, setting the detection threshold of the sensors to respond only to the retroreflective strips violates the requirement for high excess gain. In reality, the critical threshold setting required is likely to be impossible to achieve; the robot may pass through the opening closer to one side than the other, and the distances involved can vary as well due to different doorway widths ranging anywhere from 33 inches to 72 inches or more.

Even if the leading edges could be precisely detected, ambiguities arise in measuring the distance between the actual locations of the left and right targets using ultrasonic ranging techniques. Referring now to figure 83, we see that both the door stops as well as the actual door itself can interfere with the ranging process. The resulting measurement accuracy, although acceptable for determining the lateral position of the robot in the doorway, would be insufficient for the desired final resolution in heading.

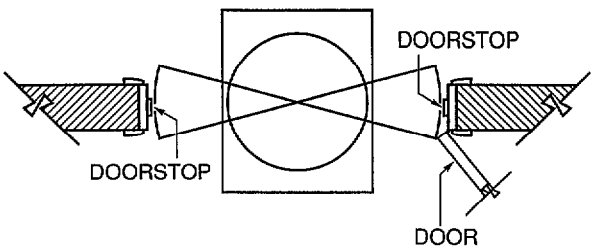


Figure 83. Both doorstops as well as the actual door itself can interfere with the ranging process.

The solution to these problems is to reorient the proximity sensors to where the beams are vertical as opposed to horizontal, yet still orthogonal to the direction of robot motion, as shown in figure 84. The target separation distance d now becomes a constant which is precisely determined by and equal to the sensor displacement l onboard the robot, eliminating one of the above concerns altogether. The upper door casing now becomes the target, where there is much less possibility of obstructions being present that might interfere with leading-edge detection.

To further address this issue, the proximity sensors can be configured in the convergent mode as opposed to diffuse mode (Everett, 1988b; Banner, 1989), taking advantage of the fact that the distance to the overhead casing will be fairly constant, regardless of the path followed by the robot through the doorway. (Standard door height is 80 inches.) This means objects (such as a ceiling or overhead light fixture) outside of the zone of potential detection will be ignored, as shown in figure 85, allowing for even greater excess gain to be employed.

4.3.1.12 Guidepath Following. The most common guidepath following schemes in use today involve some type of stripe or wire guidepath permanently installed on the floor of the operating area. Specialized sensors mounted

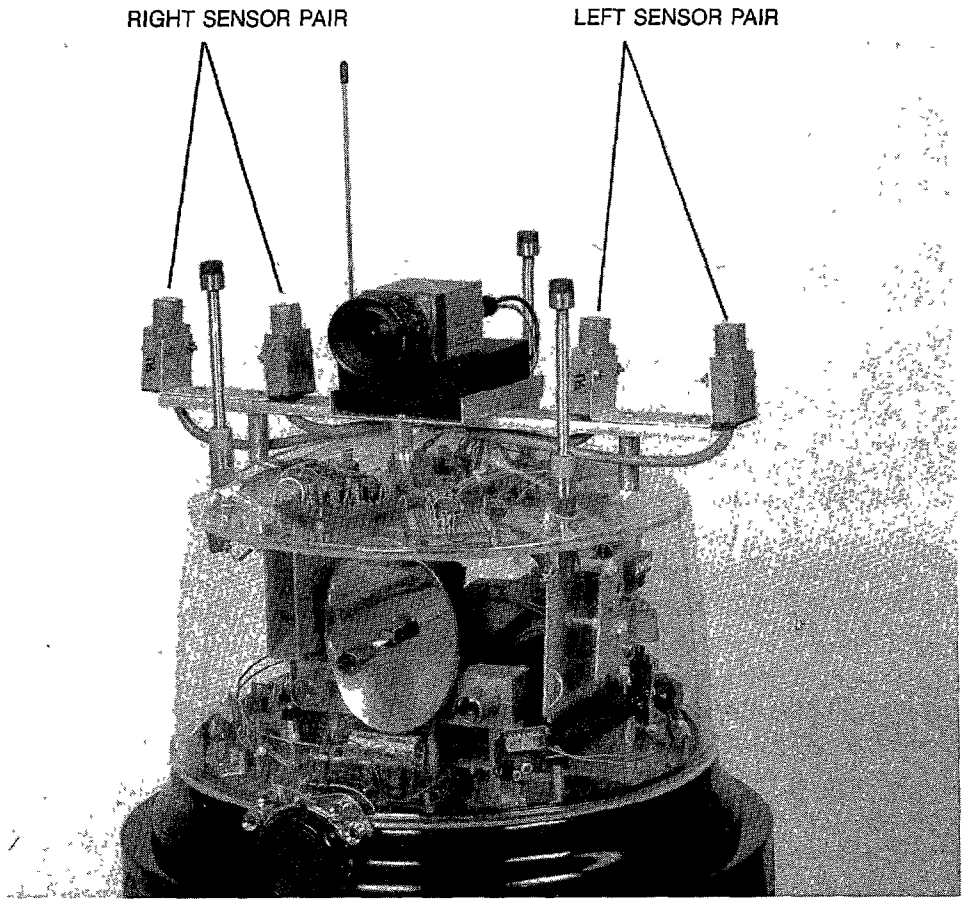


Figure 84. Vertical configuration of edge-detecting proximity sensors eliminates interference problems as well as need to measure door width.

on the front of the platform are used to servo-control the steering mechanism, causing the vehicle to follow the intended route. For purposes of this discussion, these guidance schemes can be divided into two general categories: (1) those which sense and follow the AF or RF field from a closed-loop wire embedded in the floor, and (2) those which optically sense and follow some type of stripe affixed to the floor surface. Various implementations of the latter (stripe-following) concept exist, including the most simplistic case of tracking a high-contrast (dark-on-light, light-on-dark) line, systems which track a special reflective tape illuminated by an onboard light source, and a system developed by Litton Corporation which tracks a chemical stripe that glows when irradiated by ultraviolet energy.

The prototype stripe follower developed in conjunction with this research is based on an off-the-shelf near-infrared analog proximity sensor module (P/N C5-1DN05) manufactured by Banner Engineering (Banner, 1989), used in conjunction with a 1 inch wide retroreflective tape (P/N BRT-THG-1-100). The system consists of a number (currently four) of these modules arranged in an array to yield a 4-inch footprint of illumination on the floor, as shown in the block diagram of figure 86. All sensor modules are active when the system is in the acquisition mode looking for the stripe, but only the center two modules are used once the stripe has been located and the system is in the tracking mode and centered on the guidepath.

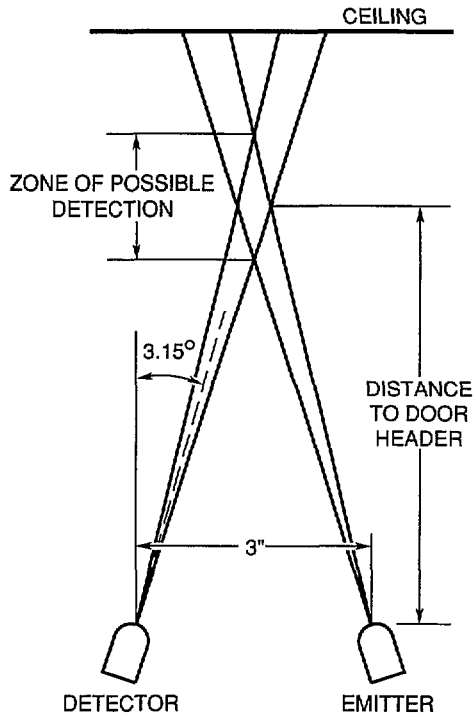


Figure 85. Objects (such as the ceiling) outside the zone of possible detection will be ignored, allowing for even greater excess gain to be employed.

The stripe follower subsystem is activated by the Scheduler, which provides power to the active near-infrared proximity sensor modules. The stripe follower subsystem then enters the *stripe acquisition* mode and begins searching for the stripe, switching to the *stripe following* mode upon command from the Scheduler once the stripe has been located. In the *following* mode, the outer two analog sensor modules are turned off to save power, and stripe offset information is calculated by the Scheduler in real time.

4.3.2 Hybrid Navigation Scheme

To preserve the validity of the world model, it is necessary to provide the system with periodic position updates which accurately reset the absolute position and orientation of the robot. In addition, autonomous transit through a congested room is somewhat slower than is desirable in some applications, in that the robot must feel its way around newly discovered transient objects. The hybrid navigational scheme overcomes these two drawbacks while retaining the

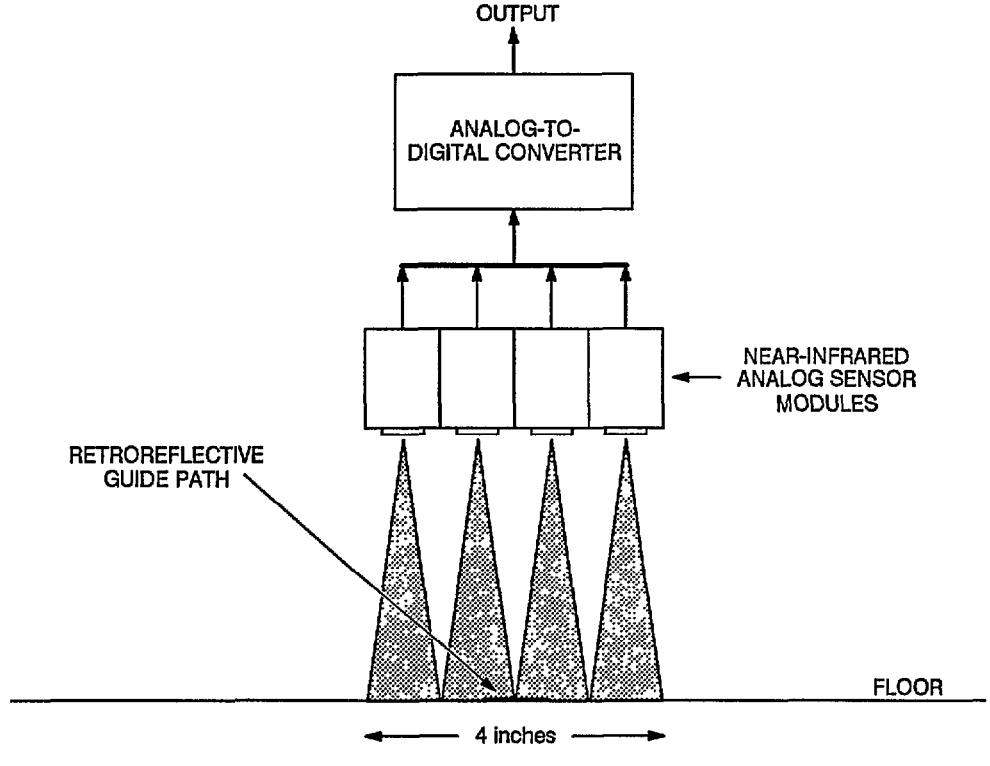


Figure 86. Block diagram of the prototype stripe following subsystem.

free-roaming flexibility of unrestricted path planner control by merging elements of reflexive control into the system. Certain highly-traveled runs (i.e., a straight-line run down a long hallway, or through the center of a large warehouse) are designated as *freeways* and marked accordingly with some type of guidepath stripe as is commonly used by AGVs.

The robot traverses this path, which is kept relatively obstacle free, at significantly higher speeds than typically possible in the *unrestricted* mode. The absolute world-coordinate position and orientation of the path is known by the planner, and the path is encoded every 3 feet to provide a position reference along the longitudinal axis as the robot moves. These specially marked locations are referred to as *exits* in keeping with the freeway analogy.

Under this scheme, the path planner calculates the nearest intercept with the freeway in planning a route to a given destination. The robot then moves in the autonomous mode to intercept the freeway, at which point a transition is made to guidepath control. The robot travels down the freeway to the exit determined by the planner to be most appropriate for the goal position. At the appropriate exit, the robot resets its position and heading to the coordinates of the exit, loads the appropriate map, leaves the freeway and resumes autonomous transit. Each time the robot returns to the freeway at various points in the course of normal operations, the dead-reckoning system is reset to preserve the accuracy of the modeling which takes place when in the autonomous mode.

As shown in figure 87, the system consists of the following subcomponents and their associated functions:

- **Planner**

  Builds and maintains the world model

  Performs path planning to generate initial route

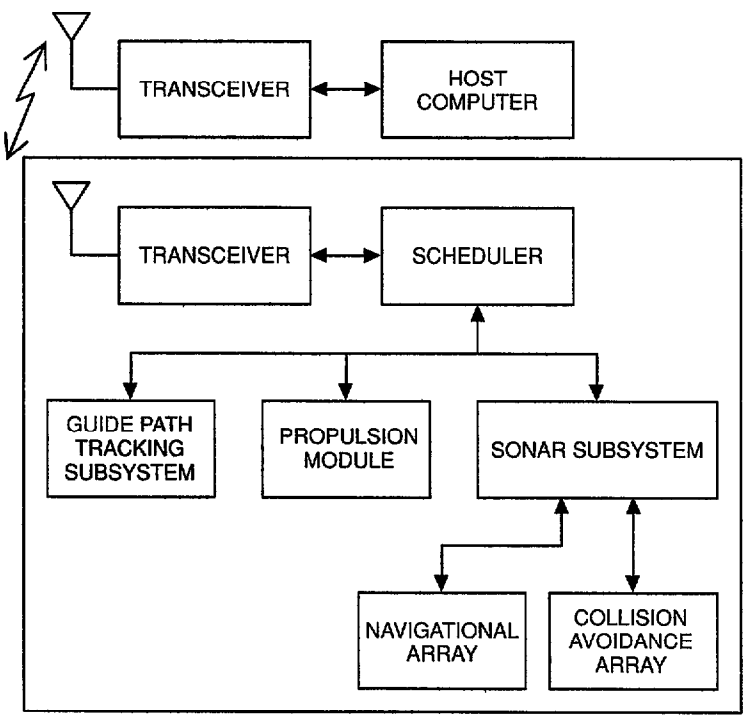


Figure 87. Block diagram of the hybrid navigational system.

Reroutes vehicle for collision avoidance

Provides an operator interface if desired

- **Scheduler**

  Receives high level instructions from Planner

  Coordinates onboard activities of all subsystems

  Passes drive commands to propulsion module

  Receives X-Y position, heading from drive controller

  Receives range and bearing data from sonar

  Checks for potential collision condition

  Sends stop command to propulsion if collision imminent

  Receives guidepath data from stripe follower

  Checks for stripe present

  Calculates steering correction for drive controller

  Checks for presence of displacement marker

  Resets dead-reckoning registers

  Passes required information back to Planner

- **Propulsion Subsystem**

  Executes movement commands from Scheduler

  Performs dead-reckoning calculations

- **Sonar Subsystem**

  Receives commands from Scheduler

  Provides ranges and bearings to nearby obstacles

- **Stripe Follower Subsystem**

  Detects guidepath

  Detects displacement markers

  Passes information back to Scheduler

4.3.3 Operational Scenario

The following operational scenario is intended to illustrate the manner in which the various subsystems involved in the hybrid navigational system interact. The robot is initially located at point A in room 101 of building F-36, as shown in figure 88. Freeway guidepath stripes running east-west and north-south, as well as numerous doorway penetration stripes, are encoded in the model. The Planner is tasked with directing the robot from its current position A in room 101 to point J in room 108, and then back again to the starting point A.

The path planner initiates an A* search originating at point A and expanding in the general direction of the destination point J, as shown now in figure 89. As the expansion routine encounters the doorway penetration stripe just after point B, the A* search reacts to the zero cost associated with such guidepath stripes, and the resulting path segment is made to precisely overlay the location of the doorway penetration stripe. Upon reaching the end of this stripe, the expansion routine continues to point C, where a change in path direction takes place as influenced by the lower perceived cost associated with expansion north towards the destination point J.

This expansion continues until the east-west freeway guidepath stripe is encountered, whereupon the expansion routine follows the zero-cost freeway to point G. The A* search continues in this fashion, following the doorway penetration

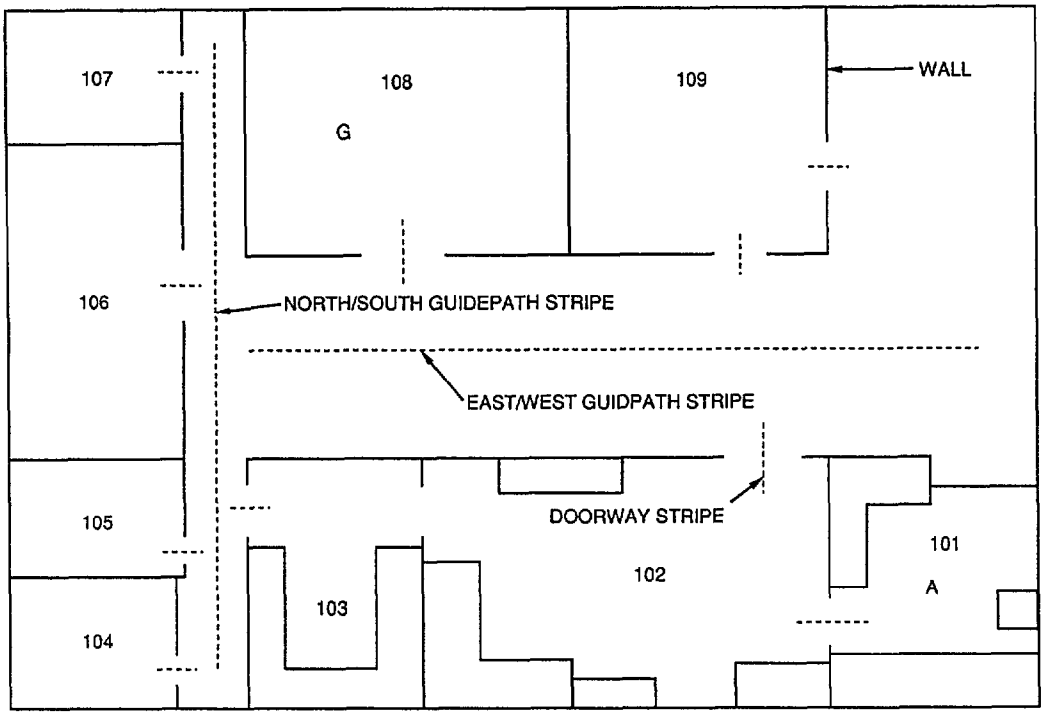


Figure 88. Map of Building F-36 showing freeway guidepath stripes.

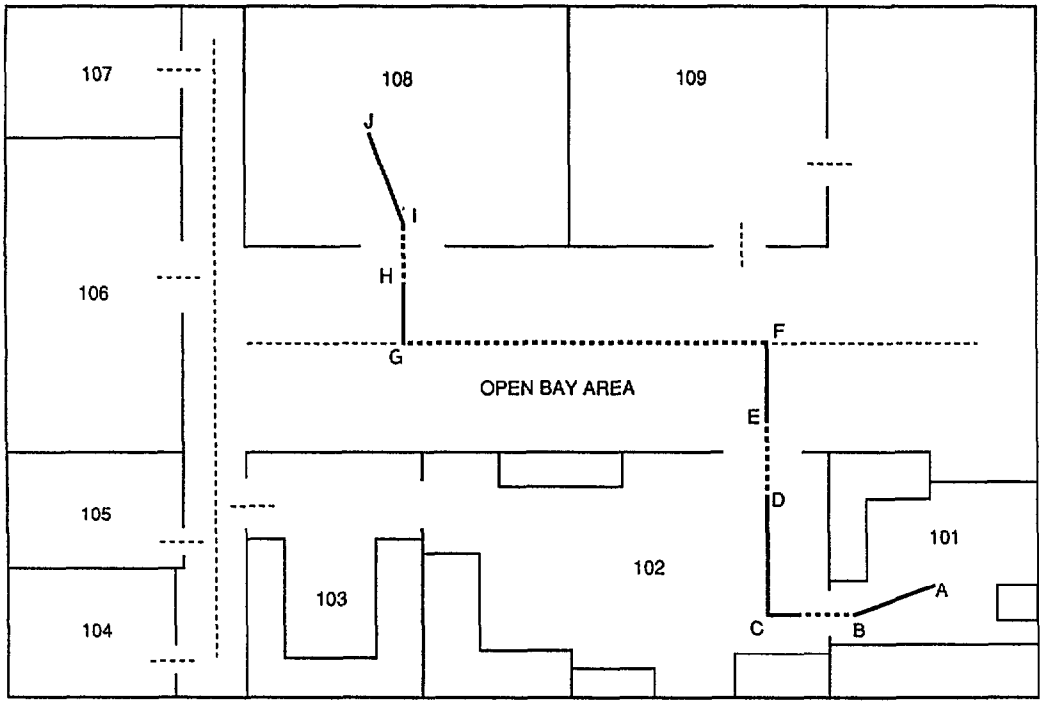


Figure 89. A* search expansion is shown as the Planner searches for the destination.

stripe at point H into room 108 at point I, and then in the most direct fashion to destination point J. The path planner then backtracks along this search route to create the list of path segments which describe the resultant found path. Appropriate movement commands are then generated by the Planner, and the first of these is downloaded via the RF link to the Scheduler onboard the robot.

Upon receipt of this move command, the Scheduler directs the drive controller (CPU #4) to move forward at the specified velocity for the specified distance (calculated by the path planner) to position the robot at point B. In responding to this command, the drive controller begins to periodically pass updated heading as well as X-Y position data calculated by its onboard dead reckoning software to the Scheduler. The Scheduler meanwhile instructs the sonar controller (CPU #3) to fire the center five transducers in the collision avoidance array in a repetitive fashion, and examines the resulting range values for a specified minimum threshold indicative of a potential collision. A halt command is issued to the drive controller in the event such a condition is found to exist. The Scheduler relays all sonar range readings and the associated dead-reckoning X-Y and θ parameters to the Planner for subsequent entry of robot and/or obstacle location information into the world model. This process continues until the robot has moved the specified distance, whereupon the Planner is informed the ordered move has been completed.

The Planner at this point issues the next move command, which instructs the Scheduler to rotate the robot to the desired new heading, as dictated by the orientation of the next path segment BC. The Scheduler relays updated heading information to the Planner during the course of this action, and informs the Planner when the move is complete. The Planner then downloads the next move command, which tells the Scheduler how far to travel along path segment BC, and in addition, informs the Scheduler the current segment BC contains a guidepath stripe to assist in doorway penetration. The Scheduler issues the move command, and then activates the guidepath tracking subsystem in the *stripe acquisition* mode. The guidepath tracking subsystem informs the Scheduler of stripe lateral offset when a guidepath stripe has been detected. The Scheduler then switches the guidepath tracking subsystem to the *stripe following* mode, and relays stripe position parameters, as well as sonar range and dead-reckoning updates, to the Planner.

The Scheduler monitors the lateral position of the stripe, and adjusts the direction of travel accordingly by way of heading adjustment commands to the drive controller, to keep the stripe centered. Once the system has stabilized with the stripe centered in the field of view of the tracking system, the Scheduler resets the vehicle heading θ to the known orientation of the guidepath stripe, and the lateral position of the vehicle to the known position of the guidepath stripe. Upon receipt of a displacement marker report from the guidepath tracking subsystem, the Scheduler also resets the longitudinal position coordinate to the known position of the marker. In this fashion, dead-reckoning errors accumulated while traversing path segment BC are canceled as the vehicle's perceived X-Y and θ parameters are reset to the known parameters of the stripe. When the doorway penetration guidepath stripe ends, the guidepath tracking subsystem will inform the Scheduler, which then shuts off guidepath tracking subsystem. No further adjustments are made to vehicle heading for the duration of the current path segment. Upon arrival at point C, the Scheduler informs the Planner, which downloads the required turn information to bring the vehicle to the appropriate heading for traversal of path segment CDEF.

This next segment is executed in similar fashion, with the Scheduler activating the tracking system at the appropriate time in the vicinity of point D to detect the doorway penetration stripe. Accumulated dead reckoning errors are again eliminated as the Scheduler resets onboard position and heading parameters to those of the stripe, and relays the same information to the Planner. Upon termination of the

stripe at E, the guidepath tracking system is shut down, and the robot travels under dead-reckoning control the remaining distance to point F.

The Scheduler next turns the robot to the new heading provided by the Planner at point F in order to traverse the east-west freeway along path segment FG. The guidepath tracking system is again activated, and the freeway stripe acquired. The Scheduler then switches to the reflexive control mode, and issues heading correction commands to the drive controller based on the output of the tracking subsystem, so as to precisely follow the freeway guidepath stripe. Lateral position and heading are reset to the known values associated with the freeway, and longitudinal position is updated each time a displacement marker is encountered. Upon reaching point G, the stripe following subsystem is deactivated, and the Scheduler informs the Planner the current move is completed.

This process of switching periodically to reflexive control for purposes of following the stripe and updating the dead reckoning position is repeated as described until the robot reaches the destination J. If so desired, the Planner can then direct the robot to return to the starting position A, and a similar procedure would be followed to retrace the route back to room 101. The point to note is after executing this round trip as described above under direction of the hybrid navigational control scheme, the accumulated dead-reckoning error at point A would be only that which accrued from point B to point A, a distance of approximately 4 feet in the example shown. Under a conventional dead-reckoning approach, the accumulated dead reckoning error would have built up over the entire route (ABCDEFGHIJ) and back (JIHGFEDCBA), a distance of approximately 130 feet. This latter distance would exceed the acceptable limits for an uncorrected move, in that approximate dead-reckoning accuracies for common propulsion subsystems are on the order of 99.5 percent of distance traveled.

5.0 INTELLIGENT SECURITY ASSESSMENT

There are many advantages afforded by the use of robotic technology in a physical security and surveillance role. Well known in the industry are the statistics on the expense of maintaining a security presence through conventional means. Average yearly costs for a single security guard on an around-the-clock watch is in the neighborhood of between $80,000 and $200,000. It has been estimated that a robotic system replacing four to five people on a 24-hour-a-day basis could achieve an annual savings of $300,000, when the issues of training, supervision and benefits are taken into account, yet some commercial facilities employ over 700 guards at a single site. The advantages of a system that will not tire, become distracted, frightened, or even subversive are obvious and well touted. The concept of mobility provides for an unpredictable pattern with respect to precise sensor location and orientation, making the job of casing a scenario for the purpose of identifying blind spots rather difficult. The random patrols and the uncertainty in the mind of the intruder as to what the robot's response might be upon detection adds a slight psychological advantage to the deterrent function.

Less obvious are the advantages associated with the realtime analysis of a tremendous amount of information in reacting to stimuli, with more *presence of mind* and less likelihood of distraction than can be expected from a human counterpart. The assurance of an orderly execution of crucial events following detection of an unwanted condition can be attractive indeed in certain critical scenarios.

A computer-based system can be equipped with the appropriate sensors to emulate all of the security perception functions of its human counterpart. Deviations from expected norms can be detected immediately, if the system were told ahead of time what conditions to monitor, and provided with the associated alert

thresholds. The human, on the other hand, is somewhat slower in response, and not always as alert. In addition, the human, unless augmented with specific electronic detection devices, is not as capable in his or her sensory perception, particularly in the case of visual obscurants and distractions. The human does, however, have a distinct advantage in recognizing and classifying stimuli which had not been previously encountered or anticipated, relying on inherent powers of association and deductive reasoning, which current computer-based systems lack. However, all such unforeseen occurrences, if detected by the robot, could be classified as possibilities for further investigation.

There exists a tradeoff point wherein it becomes more cost effective to outfit a platform with numerous sensors, and let it travel from zone to zone, as opposed to installing, wiring, and protecting the fixed installation of these same sensors so all areas within a space are effectively covered. This tradeoff is obviously a function of the conditions to be monitored and the unit cost of the appropriate sensors, as well as the area and geometric configuration of the space to be protected. Some provision must be made to preclude site vulnerability when the robot is not on station, or perhaps even down for maintenance. For these reasons, a combination of both fixed and mobile sensors is likely to evolve as the appropriate solution in most cases, but obvious problems arise with the operation of a mobile platform in an area protected by permanently installed motion detectors.

The traditional problem encountered in applying off-the-shelf intrusion sensors in an automated security system has been as the detector sensitivity is raised to provide the necessary high probability of detection, there is a corresponding unacceptable increase in the nuisance alarm rate. Operators quickly lose confidence in such a system where sensors are prone to false activation. As an example, passive infrared motion detectors can be falsely triggered by any occurrence which causes a localized and sudden change in ambient temperature within the sensor's coverage area. This false triggering can sometimes occur naturally, as in the case of the starting or stopping of a heating or cooling system. Optical motion detectors can be activated by any situation which causes a change in ambient light level. Again, this situation could be caused by some noncritical event, such as passing automobile headlights, or lightning flashes. Discriminatory hearing sensors could be triggered by loud noises originating outside the protected area, such as thunder, passing traffic, or overflying aircraft. Microwave motion detectors can respond to rotating or vibrating equipment, and so forth.

A truly robust robotic or automated security system must employ a variety of intrusion detection sensors, and not rely on any single method. This redundancy thwarts attempts to defeat the system in that there is a much higher probability of detection with multiple sensors of different types. Equally as important, such redundancy provides a means of verification to reduce the occurrence of nuisance alarms (redundant intrusion detection schemes operating on different principles will not all respond to the same spurious interference). A verification algorithm can be implemented which reacts to the first indication of a potential alert by querying other sensors for secondary indications. If the disturbance is not confirmed, the system could remain in an enhanced alert mode for a period of time, perhaps move to a better vantage point, and continue to monitor the situation. Additional active sensors could be brought into play during this period. If the presence of an intruder is not established, the system downgrades the alert condition and returns to normal surveillance (Everett, 1988a).

The strategy employed on ROBART II (figures 90a and 90b) involves using numerous types of broad coverage sensors, which typically are more energy efficient, as primary detection devices, and higher-resolution units in a secondary confirmation mode to verify and more clearly characterize a suspected disturbance. The sentry is alert at all times, but its acuity can be enhanced by self-generated actions which activate these additional systems when needed to better discriminate among and between stimuli.

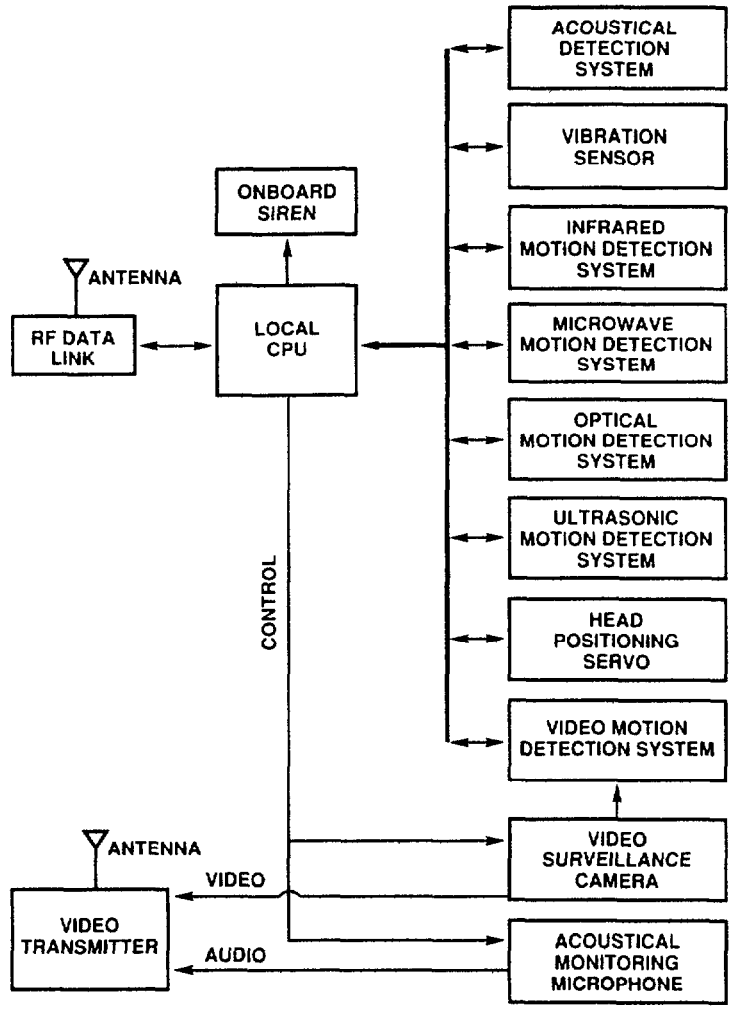


Figure 90a. Block diagram of the intelligent security assessment system onboard ROBART II.

A fixed array of sensitive, low-resolution sensors is employed with overlapping 180-degree coverage to obtain the necessary high probability of detection. The area of coverage is divided into discrete zones (figure 91), with different types of redundant motion detection schemes assigned to each zone. An array of 24 ultrasonic ranging units with 360-degree coverage can be activated to establish the position of a moving intruder with respect to the robot. A miniature high-resolution CCD surveillance camera is deployed on a panning mechanism for specific direction at areas of suspected disturbance, for purposes of verification. Assessment of the results is performed by appropriate software which cross-correlates among redundant primary sensors within a specific detection zone, and schedules and interprets subsequent verification by the secondary high-resolution sensors.

The goal of the intelligent assessment software is to make the robot sensitive enough to detect any intrusion, yet smart enough to filter out nuisance alarms (Everett, Gilbreath, and Bianchini, 1988; Everett and Bianchini, 1987; Everett, Gilbreath, Alderson, Priebe, and Marchette, 1988). A potential intrusion is more easily recognized through the use of some type of motion detection scheme, and several exist. Passive motion detectors for the most part sense a

change in ambient conditions due to movement within their field of view. This change could be associated with the observed level of illumination, thermal energy, noise, or even vibration normally present in an unoccupied space. Active detectors provide a controlled energy input into the observed environment, reacting to changes with respect to a fixed reference as caused by perturbations within the area of coverage. For this reason, active detectors can sometimes be tailored to provide more sensitivity or selectivity in a specific situation (Everett, 1988a).

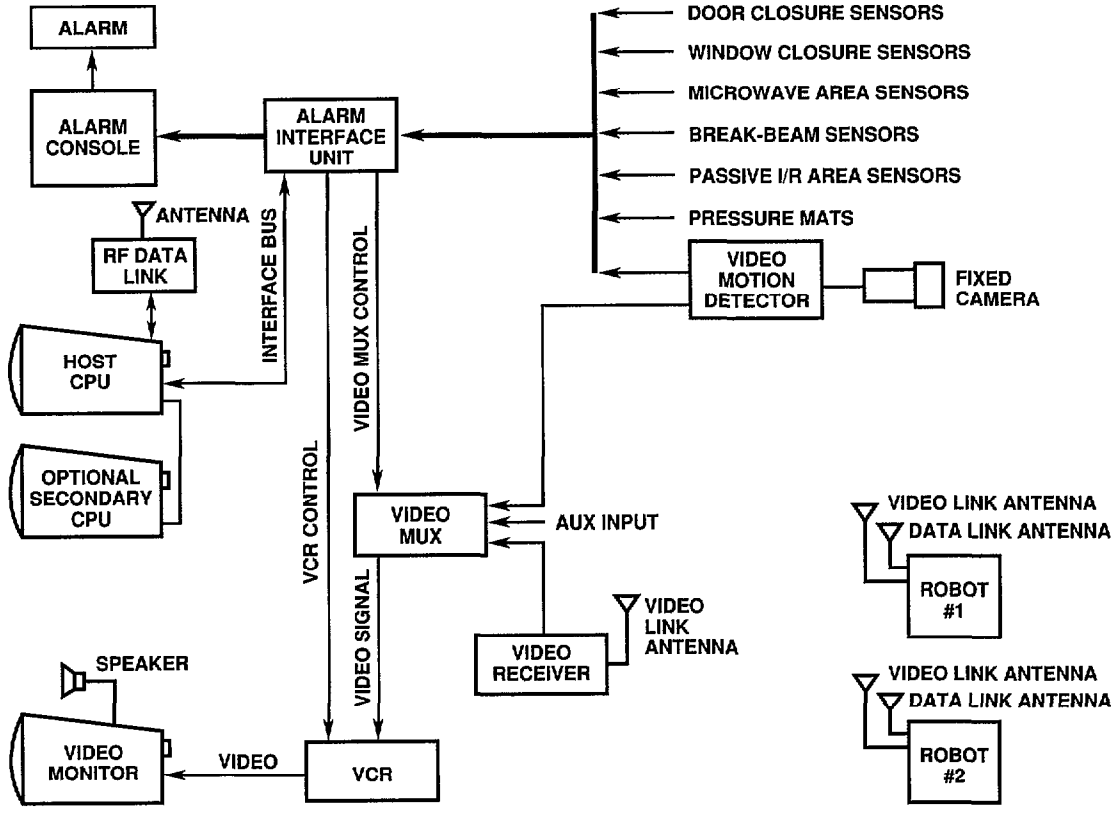


Figure 90b. Block diagram of the global security assessment system which integrates fixed and mobile sensor inputs.

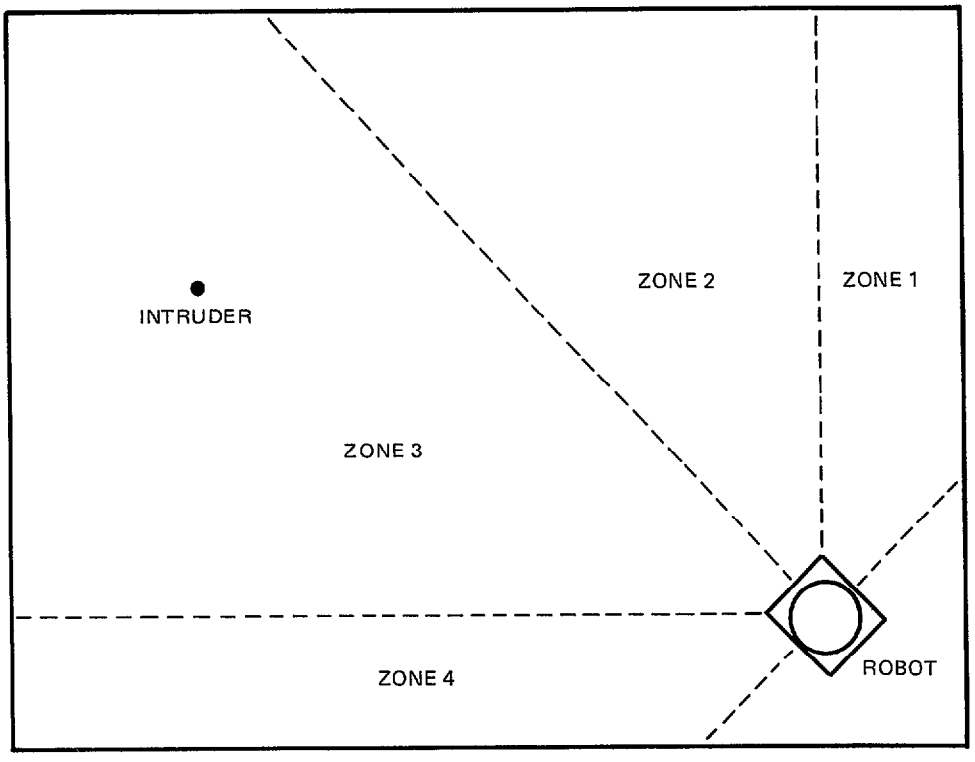


Figure 91. The six groups of intrusion detection sensors are arranged with full 180-degree coverage divided into four fan-shaped zones.

5.1 MODELING THE SECURITY ENVIRONMENT

The intent is to provide a robust scheme which allows one or more mobile security robots, each equipped with a multitude of intrusion detection sensors, to operate within a secure environment which is protected by fixed-installation alarms. The integration is accomplished in such a fashion that the robot's motion through the area does not trigger a system alarm. Data from both *fixed* and *mobile* sensors can be collectively assessed in calculating a global composite threat score from individual sensor weightings, to achieve a high probability of detection with a corresponding low nuisance alarm rate.

In order to pursue the optimal mix of fixed and mobile sensors, and provide the intelligence necessary for a computer-based system to emulate the assessment functions of its human counterpart, some method of data fusion must be achieved. The system must know at all times where the robot is located, the zones of coverage for its onboard sensor suite, and the resultant effect of its presence or motion on fixed intrusion detection sensors viewing that same area. These needs are addressed through implementation of a global world model representing the total area under surveillance, as will be discussed below.

The world model consists of a number of bit-mapped parallel arrays, or layers, each indexed to an absolute X-Y grid of the floor plan. The floor plan is typically divided up into a number of subset floor plans to achieve a realistically sized model in order to facilitate near realtime manipulation of the encoded information. For any given subset floor plan, the first layer is devoted to X-Y positional information used for navigation and collision avoidance, as previously described in section 4.2.

Two additional parallel arrays, or layers, are assigned to the world model and used to represent the areas of coverage of: (1) the fixed-installation security sensors for that portion of the floor plan, and (2) the mobile security sensors mounted on the robot. These additional layers will be referred to as the *fixed* sensor coverage layer, and the *mobile* sensor coverage layer, as shown in figure 92.

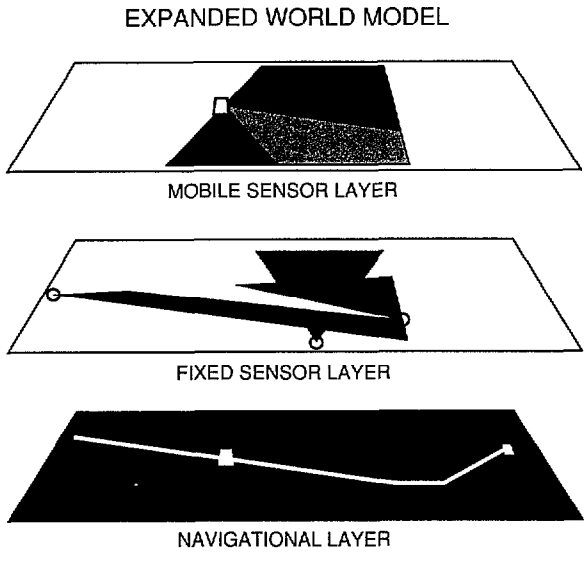


Figure 92. The expanded world model employs two additional layers to represent the coverage areas of the fixed and mobile intrusion detection sensors.

Since the robot by design is to move about the area under surveillance in a patrolling fashion, its position and orientation must be taken into account when calculating the absolute map representations of the onboard security sensor coverage areas. The representations must be translated and rotated in accordance with the robot's motion. The resulting *mobile* layer bit values can then be fused with those from the *fixed* sensor layer for those sensors which are triggered, and the result stored in a fourth layer which reflects the integration. This fourth layer will be referred to as the *hit* layer.

This research thus extends the concept of a robotic security system to include the tasks of verification and assessment (as opposed to merely detection). Provision must be made for dealing with aberrations in the sensors themselves giving rise to erroneous data, spurious readings due to naturally occurring external events, and failure of a discrete sensor or even subsystem. The problem can be subdivided into the two areas of realtime assessment of instantaneous sensor data, and secondary assessment of historical sensor data.

5.2 REALTIME ASSESSMENT SOFTWARE

The realtime assessment software monitors the instantaneous state of each of the fixed and mobile security sensors to determine the presence or absence of an intruder. Cross correlation of the mobile sensor outputs takes place locally onboard the robot at the Scheduler level, and is referred to as Local Security Assessment (section 5.2.1). Results are transmitted over the RF link to the host computer for further analysis, data logging, and console display as appropriate. A higher level correlation can then be performed at the Host, which takes into account the fixed installation sensors, as well as reports from other mobile robots operating in the area. This higher level correlation is referred to as Global Security Assessment (section 5.2.2).

5.2.1 Local Security Assessment

Seven different types of sensors are used onboard ROBART II (infrared, microwave, ultrasonic, optical, video, sound, and vibration) to both increase the likelihood of detection and decrease the frequency of nuisance alarms. Figure 93 shows a sample screen display. The upper half of the display shows the state of each of the sensors, the bearing to a possible intruder, and the current alarm state. Sensors with a black background are temporarily disabled or unavailable. The lower half of the screen is used for displaying robot status information and the current environmental conditions.

The motion detection sensors (depicted in the upper half of the display) are grouped into four zones, each zone containing several different types of sensors. The software performs a summation of weighted scores for all sensors within a particular zone, and calculates a composite threat score (shown in the *ALARM* box in the upper right of figure 93) which is proportional to the perceived threat presence. The individual sensor weights are initially established through statistical analysis of data characterizing sensor performance under known conditions as logged over a long period of time.

The software detects patterns, such as purposeful motion across adjacent zones, and increases the associated composite threat accordingly. The realtime assessment software then activates and positions secondary verification sensors as needed. At the same time, the current alarm threshold is dynamically calculated, based on the number of sensor groups which are available, and other relevant conditions, such as ambient lighting, time of day, etc. The system classifies an alarm as an actual intrusion only when a complete evaluation has been performed using all sensor groups, and the resulting composite threat score exceeds the alarm threshold.

5.2.2 Global Security Assessment

The Global Security Assessment software must address two fundamental issues: (1) inhibiting those fixed-installation sensors which are momentarily activated by the robot's passage through the protected area, and (2) fusing the alarm status data from the fixed-installation sensors with the data from the mobile sensors mounted on the robot (or robots) in order to create a composite representation of the perceived threat.

5.2.2.1 Inhibiting Fixed Sensors. When the robot is moving in a room which is protected by fixed-installation sensors, the security assessment system must determine when the robot is about to enter or leave the coverage area of any given sensor. The sensor in question must be inhibited for that portion of the time when the robot is moving through its field-of-view, so the motion of the robot does not generate a nuisance alarm. Once the robot has departed the coverage area, that particular sensor must be re-enabled.

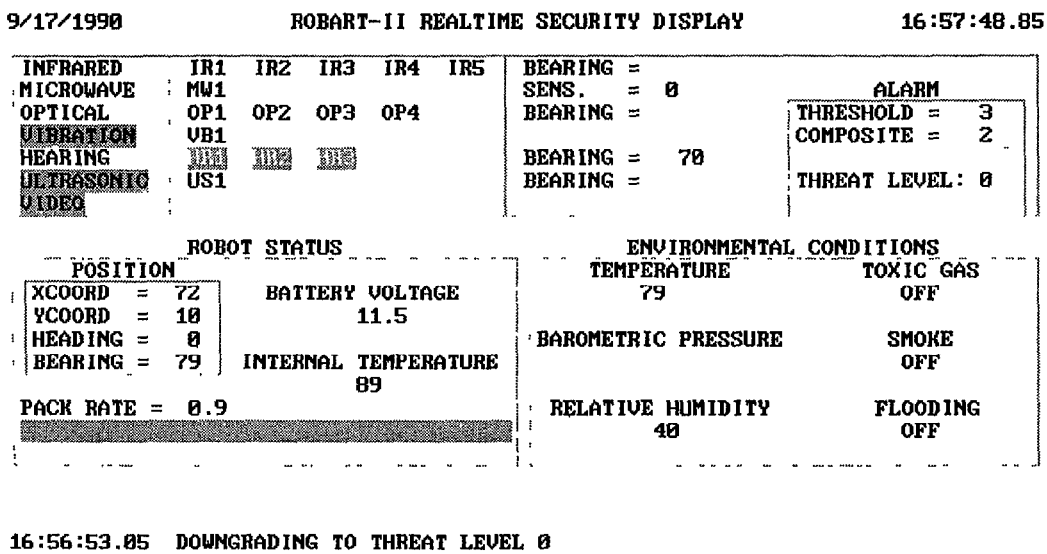


Figure 93. In the upper left-hand window of the Security Display, sensor groups which are not currently available are depicted in reverse video. Just to the right, individual sensors within the active groups are portrayed in reverse video when alarmed.

The robot's X-Y position can be used to obtain from the *fixed* sensor coverage layer the ID of all fixed sensors that could be affected by the robot's motion at that particular instant. These sensors are then gated out when constructing the associated *hit* layer which is used to calculate the global composite threat. Note the robot's *mobile* coverage layer would be different when the robot is in motion, generated under a second set of rules that applied to the case of a moving platform.

With the bitmap approach, it is only necessary to see if any point in the *fixed* sensor coverage layer lying within in a small area of ambiguity around the robot has the appropriate bit set for that particular sensor. At the beginning of each inhibition loop, the security assessment software must mark all the sensors as uninhibited, then inhibit only those affected by the robot. This is so that when a robot leaves the coverage area, the associated sensor will be enabled once again.

Alternatively, this inhibiting function can be accomplished in a fashion other than the bitmap approach by examining each sensor coverage polygon (predefined by the operator using the Map Editor) to see if the robot is within that polygon, then inhibiting those sensors affected by the robot's presence. If the robot is modeled as a simple convex polygon (e.g., a rectangle), then the Sutherland and Hodgman (1974) polygon clipping algorithm can be used to determine whether or not the robot is completely outside or partially or completely inside the defined region. This particular technique will clip a concave or convex polygon (sensor coverage polygon in this case) to an arbitrary convex polygon (robot). Each sensor area touched by the robot is inhibited, and those not touched are enabled. If both the coverage and robot polygons are convex, then the Cyrus and Beck (1978) algorithm can be used.

5.2.2.2 Correlation of Fixed and Mobile Sensors. Once the fixed sensors affected by the robot have been inhibited, it is necessary to correlate all the fixed sensors with each other, as well as with the mobile sensors on the robot. Three different approaches may be used: (1) if all sensors are modeled with bitmaps, then logically AND all the maps together to form a *hit* map; (2) if all the sensors are modeled solely with polygons, then it is necessary to determine a specific polygon that is the largest subset of all the active polygons; (3) a hybrid scheme can be employed which forms a *hit* bitmap based on how many polygons cover each cell in the map. In all cases, it is first necessary to rotate the robot's coordinate system to correspond with that of the fixed sensors.

5.2.2.2.1 Bit Map Correlation. When the robot is stationary in the Security Assessment Mode, the system must determine the degree of interaction of sensors on the robot with fixed sensors in the area for composite threat weighting purposes. Accordingly, the system first determines the *fixed* sensors which are active and their associated coverage areas as stored in the *fixed* coverage layer. This information is then logically ANDed with the calculated coverage areas, in absolute coordinates, of the active robot sensors. The result is stored temporarily in a *hit* layer, which is then used to calculate the global composite threat. The manner in which this calculation is accomplished is discussed in the following paragraphs.

To initially set the appropriate bits in the *fixed* sensor coverage layer, an operator would at the time of system installation use the Map Editor and draw in the coverage extents for the designated sensor after marking its physical location in the map. An associated post-processing routine uses (upon exit from the Map Edit mode) a standard polygon scan-conversion algorithm (Rogers, 1985) to set the correct bit for that particular sensor. The assigned bit corresponds to the sensor ID (0 to 7), which is printed on the screen when the user places the Map Editor cursor on the pixel which marks the sensor location. Once this process is completed for all the *fixed* installation intrusion sensors, the *fixed* sensor array retains these values until modified by the user at a later date.

The *mobile* sensor coverage layer associated with the robot, however, is not generated in advance, to avoid the computationally expensive

array rotation problem. Instead, the *mobile* layer is created only when needed to fuse data between the fixed sensors and the robot's sensors. The information on robot heading and X-Y position is first obtained from the dead-reckoning software module, and then the *mobile* array values are generated according to pre-established rules (discussed later), taking into consideration the afore-mentioned position parameters. Since there are more than eight sensors on the robot, sensor zones (composite sensors) versus discrete sensors are used in the representation. A 16-bit representation could be employed in similar fashion if necessary to overcome this limitation.

The first four zones are as depicted in figure 92, and represent the passive infrared detectors ORed with the optical motion detectors (four zones representing eight sensors). This diagram illustrates how the zones will actually be reoriented within the layer as the robot assumes different positions within the map confines. The trick is to freeze the position and orientation of the robot before calculating the array values. Again, a standard polygon scan-conversion algorithm will set the appropriate bits for the X-Y locations in the array up to the imposed extents, which will be pre-defined as discussed below.

The four zones of figure 92 are represented as wedges out to a distance of some maximum limit, or the map extents, whichever comes first. The microwave motion detector is constrained by the room boundaries, as might be the vibration sensor; if either of these type sensors are alarmed, the corresponding bits for all array locations in the room are set. The ultrasonic motion detector is constrained to where all bits were set within a circle of ambiguity of some prespecified diameter, centered at the reported range value of the disturbance along the appropriate bearing. The limits on an acoustical hit could be defined as a wedge of some specified angle of uncertainty along the calculated bearing line, out to a distance constrained by the room or map boundary.

When a global correlation is desired in a static scenario, the Planner first runs the routine which sets all the appropriate bits in the *mobile* layer for the robot's sensor suite based on the robot's current position and heading, and in essence effects a coordinate transform which makes the mobile sensors look like fixed sensors for that point in time. Then the software looks to see which sensors are alarmed, and begins to construct the *hit* layer array using the sensor coverage information encoded in the *fixed* and *mobile* sensor layers. The *hit* layer is then used to generated the global composite threat.

5.2.2.2.2 Intersecting Polygons Correlation. The basic idea here is very simple: find the largest polygon completely contained within all the other active polygons, i.e. find the largest subset that is a subset of all other subsets. This is most easily accomplished using the following steps:

> Pick an initial polygon (any one will do).
>
> Pick another polygon that has not been clipped.
>
> Clip the two polygons against each other, resulting in a third polygon representing the intersection.
>
> Using the resulting polygon as the initial polygon, repeat the process until all polygons have been clipped.

The output is the maximum size polygon contained within all the other polygons. If all polygons are guaranteed to be convex, then the Cyrus and Beck (1978) clipping algorithm can be used. Otherwise, each concave polygon must first be decomposed into two or more concave polygons, then clipped. Alternatively, a more general algorithm capable of clipping a concave polygon against another concave polygon could be used (such as the Weiler and Atherton (1977) algorithm). One drawback of this technique is if a sensor is firing erroneously and it does not overlap each of the other sensors, then no polygon will result.

5.2.2.2.3 Hybrid Correlation. This scheme produces a bitmap as output. Each cell in the map records the number of polygons containing that particular cell, i.e., the number of *hits*. The cells are set by scan-converting each of the active polygons. Every time a cell is found to be

within a polygon, the contents of that cell are incremented. Once all the polygons have been scanned, the hit map can be used as a pseudo-probability map. Those cells with the most number of hits are most likely to be the current location of the intruder. Two techniques for the doing the scan conversion are immediately obvious, though others may be used.

The first scheme involves creating two maps, the hit map and the scan conversion map. The current polygon is first drawn into the scan conversion map, then filled using a standard raster graphics filling algorithm (Smith, 1979). Then, each cell in the hit map, whose corresponding cell in the scan conversion map is set, is then incremented. This process is repeated for all the active polygons. This requires twice the memory space, but is fundamentally very simple.

The second scheme scan-converts each polygon directly into the hit map using an ordered edge list algorithm (Rogers, 1985). Each cell found to be within the polygon is incremented.

5.3 SECONDARY ASSESSMENT SOFTWARE

The secondary assessment software runs in the background, analyzing large amounts of logged data produced by the realtime assessment software. The intent is to look for historical trends and patterns which do not show up when assessing only the instantaneous data. The secondary assessment software outputs parameters which will vary the operation of the realtime assessment loop. The initial attempt at implementation was through a neural network simulation running on a satellite 80386-based personal computer. Neural network technology was thought to add benefits such as an increase in processing speed, better fault tolerance, and the ability to continue processing despite missing or erroneous incoming data (Everett et al., 1988).

The robot currently uses a simple on/off alarm indicator based on a summed score of weighted sensor data passing a predetermined alarm threshold. This scheme, while useful and simple to implement, yields little in the way of real platform intelligence. An effort is underway which is aimed at interpreting the combinations of sensor reports as *high-level* activities. The basic premise of this network approach is that important information, useful for distinguishing which activity is occurring, can be obtained from an analysis of not only which sensors fire, but also the relative time order in which these sensors fire (Everett et al., 1988). A network architecture designed to process in the temporal domain (Priebe, 1988) has been developed and is currently being used to analyze the robot's sensor reports.

The network architecture essentially builds a pattern matcher designed to be used in a noisy environment. A supervised learning technique is employed in which all sensor reports from a continuous segment of time are correlated with a particular known activity (e.g., intruder entering the window). The network then develops *templates*, or learned patterns associated with these known activities. The templates are used to identify which, if any, of a set of known activities are occurring at the time of interest. Such a system gives the platform additional *environmental awareness* that simple on/off alarms do not provide. The system allows the robot some level of understanding about the activities occurring in the immediate environment, and is useful in many areas of evaluation and detection (Everett et al., 1988).

This increased environmental knowledge allows for a more dynamic threshold calculation. Based on the likelihood of an activity occurring at a given time, the threshold used for determining alert status may be dynamically updated. Thus the activity-dependent processing allows different levels of acuity for different situations.

Another important benefit gained from temporal processing is the detection of defective sensors. The network is dynamic, and has the ability to learn if a given sensor is either firing spuriously or not operating at all, the two most obvious potential defects in a sensor. If a sensor is firing erroneously, the system learns that the output of the defective sensor does not contribute to the analysis of the situation as does the data from those sensors in proper

working condition. For the nonoperational sensor scenario, the system must distinguish between whether the sensor is operational and not sending data, or is not operational. The scenario suggests that, given enough evidence from operational sensors, if the sensor in question were operational it would be firing, and therefore it can be assumed to be defective (Everett et al., 1988).

5.4 OPERATIONAL SCENARIO

As an example operational scenario, the robot is assigned a patrol route (or a discrete location) by the operator at a central security monitor. The robot plans a path to and assumes the first surveillance position, with all primary detection sensors online (optical, acoustical, infrared, vibration, and microwave). A half-duplex RF link is operational between the battery operated mobile platform and the 80386-based host computer at the central security monitor. The video surveillance camera and associated RF link are deactivated.

A possible disturbance is detected by one of the fixed-installation motion detectors and relayed to the host computer. The operator is alerted by a *beep* from the console. The host determines that the sensor in question could not have been set off due to the robot's motion by noting that the current dead-reckoned position is not within the designated coverage area of the sensor. The Planner therefore automatically dispatches the robot to a location where the mobile sensors can observe the area in question. With no additional confirmation from the mobile sensors, the realtime assessment software after a designated period of time classifies the threat as a nuisance alarm. If the fixed sensor continues to indicate an alarm, the offline assessment software reacts by decreasing the weighting factor for that particular sensor element, and warns the operator that the sensor may be defective. All fixed and mobile sensor reports are continuously time stamped and logged to disk for later replay and analysis. The robot then continues its assigned patrols.

A second disturbance is later detected by another of the fixed-installation sensors, and the robot is again dispatched to monitor the area and assess the situation. This time, however, the primary mobile detection sensors also react in confirmation. The threat level is sufficient for the software to activate secondary sensors, and the ultrasonic array is powered up. Cross correlation between sensors shows a strong likelihood of an intruder at position (X,Y) on the map of the secure area, now displayed on the lower half of the control screen (figure 94).

The realtime assessment software has now activated the video camera onboard the robot, and the camera is being automatically positioned by the robot to the bearing of the disturbance. The operator is notified through a second audible alert, while the video motion detector assesses the scene under surveillance. The presence of motion in the video confirms an actual intrusion, and the system sounds an alarm. The operator has the intruder in sight on the video display, sees the (X,Y) position depicted in the floor plan map on the control console, and takes the appropriate response action.

The robot meanwhile begins to follow the intruder as he moves through the building, advising him to halt. Once the relative bearing to an intruder has been established, it can be used to calculate a motion command which causes the platform to rotate in place until the disturbance is directly ahead of the robot, centered on the axis of the collision avoidance ranging array. Range information gathered by the collision sonar array, normally used to avoid an object in the path of the robot, can also be used to move towards and follow an intruder. As ranges are repeatedly obtained along fixed bearings fanning out in the direction of travel, it is a fairly simple matter to track a specified target within the field of view even while both the target and robot are in motion. The robot's

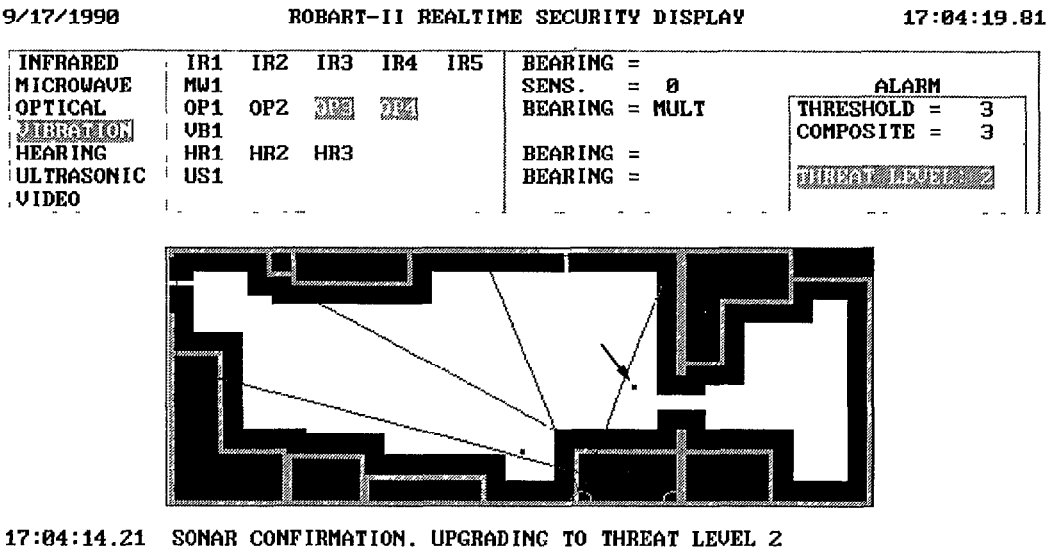


Figure 94. The position of the detected intruder (arrow) is depicted in the X-Y floorplan.

mean forward velocity is adjusted as a function of range to the target, and a calculated differential in left and right drive motor speeds is introduced as a function of how far off centerline the target appears. This differential causes the robot to turn towards the target being followed in a controlled fashion, until it appears centered, all the while maintaining a specified distance interval. The robot's absolute X-Y position is graphically displayed on a map of the workspace, while the head mounted camera transmits live video to the security display screen for evaluation by the operator.

The operator also has the choice of enabling data logging directly to the host computer hard disk of all information displayed on the security assessment screen. Significant data compression is achieved by saving only data which represents a change in a previous sensor status. Log files may be replayed, after the fact, through the same operator interface. A full complement of file commands allow the user to step through or search for various sensor conditions, or replay the files in pseudo-realtime. These statistical files are an important tool in analyzing sensor performance in intrusion detection; large amounts of data collected over an extended period of time will be analyzed to better characterize the individual intrusion detection sensors with regard to their performance under varying conditions. The resultant information will be used to develop a more intelligent algorithm for differentiating between actual and nuisance alarms.

6.0 REFERENCES

Banner Engineering Corporation. 1989. Minneapolis, MN, Photoelectric Controls and Reference Manual, P/N 04000.

Banzil, G. et al. 13 April 1981. "A Navigation Subsystem Using Ultrasonic Sensors for the Mobile Robot Hilare," *Proceedings of 1st Conference on Robot Vision and Sensory Control*, Stratford/Avon, U.K.

Beckerman, M. and E. M. Oblow. February 1988. "Treatment of Systematic Errors in the Processing of Wide Angle Sonar Sensor Data for Robotic Navigation," Oak Ridge National Laboratory Technical Memo, CESAR-88/07.

Brooks, R. A. 1983. "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Trans, on System, Man, and Cybernetics*. Volume SMC-13 No. 3.

Brooks, R. A. and T. Lozano-Perez. 1983. "A Subdivision Algorithm in Configuration Space for Findpath with Rotation," *International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany.

Crowley, J. L. 1983. "Position Estimation for an Intelligent Mobile Robot," *1983 Annual Research Review, Robotics Institute*, Carnegie Mellon University.

Crowley, J. L. March 1985. "Navigation for an Intelligent Mobile Robot," *IEEE Journal of Robotics and Automation*, Volume. RA-1, No. 1.

Cyrus, M. and J. Beck. 1978. "Generalized Two- and Three-Dimensional Clipping," *Computers & Graphics*, Vol. 3, pp. 23-28.

Davidovici, Sorin. June 1990. "On the Radio," *BYTE*, pp. 224a-228.

Devore, Jay L. 1982. *Probability & Statistics for Engineering and the Sciences*, pp. 422-436, Brooks/Cole Publishing Company.

Dunkin, W. M. March 1985. "Ultrasonic Position Reference Systems for an Autonomous Sentry Robot and a Robot Manipulator Arm," Masters Thesis, Naval Postgraduate School, Monterey, CA.

Everett, H. R. October 1982. "A Microprocessor Controlled Autonomous Sentry Robot," Masters Thesis, Naval Postgraduate School, Monterey, CA.

Everett, H. R. July 1985. "A Multi-Element Ultrasonic Ranging Array," *ROBOTICS AGE*, pp. 13-20.

Everett, H. R. March 1988a. "Security and Sentry Robots," *International Encyclopedia of Robotics Applications And Automation*, John Wiley.

Everett, H. R. March 1988b. "Survey of Collision Avoidance and Ranging Sensors for Mobile Robots," Technical Report No. 1194, Naval Ocean Systems Center, San Diego, CA.

Everett, H. R. and G. L. Bianchini. June 1987. "ROBART II; An Intelligent Security Robot," *Proceedings, U.S. Army Training and Doctrine Command Artificial Intelligence and Robotics Symposium.*

Everett, H. R. and A. M. Flynn. October 1986. "A Programmable Near-Infrared Proximity Detector for Mobile Robot Navigation," *Proceedings SPIE Mobile Robot Workshop*, Cambridge, MA.

Everett, H. R., G. A. Gilbreath, S. L. Alderson, C. Priebe, and D. Marchette, D. June 1988. "Intelligent Security Assessment for a Mobile Sentry Robot," *Proceedings, 29th Annual Meeting, Institute for Nuclear Materials Management*, Las Vegas, NV.

Everett, H. R., G. A. Gilbreath, and G. L. Bianchini. January 1988. "Environmental Modeling for a Mobile Sentry Robot," NOSC Technical Document 1230, Naval Ocean Systems Center, San Diego, CA.

Fryxell, Ronald C. 1988. "Navigation Planning Using Quadtrees," *Mobile Robots II*, William J. Wolfe, Wendell H. Chun, Editors, *Proc. SPIE 852*, pp. 256-261.

Hansford, A. July 1987. "The AD9502 Video Signal Digitizer and its Application," Analog Devices Application Note C1100-9-7/87, Norwood, MA.

Harrington, J. J. and P. R. Klarer. July 1986. "Development of a Self-Navigating Mobile Interior Robot Application as a Security Guard/Sentry," Sandia Report SAND86-0653, Sandia National Laboratories.

Harrington, J. J. and P. R. Klarer. May 1987. "SIR-1: An Autonomous Mobile Sentry Robot," Technical Report SAND87-1128, UC-15, Sandia National Laboratories.

ILC Data Device Corporation. April 1982. "Synchro Conversion Handbook," Bohemia, NY.

Kadonoff, Mark B. 1990. "Ultrasonic wall-following controller for mobile robots," *Mobile Robots IV*, William J. Wolfe, Wendell H. Chun, Editors, *Proc. SPIE 1195*, pp. 391-401.

Laird, R. T. and R. P. Smurlo. 1990. "System Specification for a Modular Robotic Architecture," NOSC Technical Document in work, Naval Ocean Systems Center, San Diego, CA.

Lee, C. Y. September 1961. "An Algorithm for Path Connections and Its Applications," *IRE Trans. Electron. Computing*, vol. EC-10, pp. 346-365.

Lozano-Perez, Tomas and M. A. Wesley. 1979. "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of the ACM*, Volume 22, Number 10, pp. 560-570.

Martin, S. W. and R. C. Hutchinson. 1989. "Low-Cost Design Alternatives for Head-Mounted Displays," *Proceedings, SPIE 1083, Three Dimensional Visualization and Display Technologies*.

Maxwell, K. 20 May 1990. "Wireless LANs," *PC Magazine*, p. 318.

Moravec, Hans P. and Alberto Elfes. March 1985. "High Resolution Maps from Wide Angle Sonar," *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, pp. 116-121.

Moser, J. and H. R. Everett. November 1989. "Wide-Angle Active Optical Triangulation Ranging System," *Proceedings of the Advances in Intelligent Robotics Systems: Mob Robots IV.*

Oetting, John D. December 1979. "A Comparison of Modulation Techniques for Digital Radio," *IEEE Trans. on Comm.*, Vol. COM-27, NO. 2, pp. 1752-1762.

Priebe, C. May 1988. "Temporal Information Processing," Master's Thesis, San Diego State University.

Rathbone, R. R., R. A. Valley, and P. J. Kindlmann. December 1986. "Beacon-Referenced Dead Reckoning: A Versatile Guidance System," *Robotics Engineering*.

Rogers, David F. 1985. "Procedural Elements for Computer Graphics," McGraw-Hill Book Company, New York.

Rubin, Frank. September 1974. "The Lee Path Connection Algorithm," *IEEE Transactions on Computers*, Volume C-23, No. 9, pp. 907-914 .

Sklar, Bernard. 1988. *Digital Communications*, Prentice Hall, New Jersey.

Smith, Alvy Ray. 1979. "Tint Fill," *Computer Graphics*, Vol. 13, (Proc. SIGGRAPH 79).

Stover, Dawn. April 1990. "World Without Wires," *Popular Science*.

Sutherland, Ivan, E. and Gary W. Hodgman. 1974. "Reentrant Polygon Clipping," *CACM*, Vol. 17, pp. 32-42.

Weiler, Kevin and Peter Atherton. 1977. "Hidden Surface Removal Using Polygon Area Sorting," *Computer Graphics*, Vol. 11, pp. 214-222, (Proc. SIGGRAPH 77).

Winston, P.H. 1984. *Artificial Intelligence*, Addison-Wesley, Reading, MA, pp.101-114.

Ziemer, Rodger E. and Rodger L. Peterson. 1985. *Digital Communications and Spread Spectrum Systems*, Macmillan Publishing Company, New York.

APPENDIX A

HIGH LEVEL SOFTWARE DESCRIPTION

This appendix describes ROBART's higher level software, which runs on an 80386-based PC/AT with the following hardware configuration:

(1) 80386 PC-XT/AT Host CPU

* 640K RAM
* EGA card
* COM1 connected to ROBART
* COM2 connected to item 2) below (optional)

(2) Optional 80386 Offline Assessment CPU

* COM1 connected to COM2 of Host item 1) above

The software also uses in-house developed graphics serial I/O libraries. All high-level code is written in Microsoft C V6.00 using the large memory model.

The goal for the higher level software is to (1) navigate ROBART over previously explored terrain without running into fixed or transient objects, and (2) perform the security functions of intrusion detection, verification, and assessment. The software is broken down into 6 subtasks: map generation, map editing, global path planning, collision avoidance, path planning, position estimation, and intrusion detection. Each of these tasks and their proposed solutions have been described previously in the text.

The software has been developed over the past 3 years in a modular "demo" mode: each of the subtasks can be chosen separately from a main menu (see robart.c) and executed. The final version will primarily execute script files - a sequence of hight level commands to be executed sequentially by the robot. Such commands might be patrol, move to position on map, recharge, etc., (see script.c.)

Currently the software has the control structure described in figure 95. The following is a list of all the important files along with a three-part description:

(1) What the module implements

(2) Fixes that need to be made in the short term

(3) Long-range improvements

/**********Main module**********/

ROBART C 18134 08-02-90 3:35p

Contains the routine main. Sets up the display and keyboard communication. Handles the main menu and dispatches menu choices. No changes are currently planned

/**********Primary modules**********/

MAKEMAP C 35886 08-07-90 1:53p

The map maker. This module is used when the robot makes its own maps using the navigational sonar array. It also allows the operator to manually drive the robot around using the cursor keys.

Future improvements would be to make the robot driving ability available to all other modules.

PATHPLAN C 47638 08-09-90 5:30p

The path planner. The path planner takes a floor map file on input and, depending on the operator's choice of starting and destination positions, finds a path if one exists. The result is an inflection list, which is ready to be processed by other modules.

A method for removing very short line segments needs to be developed.

Future improvements might be to improve the algorithm to pick a "better" path further away from objects in the work space.

INTRUDER C 48968 08-09-90 8:05a

The intrusion detection or security assessment module. This module is a realtime display panel of various security related sensors, which also allows the logging and out-of-realtime replay of sensor data. The sonar plotting routine is overlayed on the lower half of the screen when an intruder is detected.

There should be an option to send the intrusion detection packets out the second serial port to be processed by the neural network computer.

Future improvements would be to accept data back from the neural network to update sensor weights.

SONPLOT C 24617 08-09-90 7:38a

The sonar plotting and position estimation module. The sonar plotting portion takes returns from the upper ring of sensors and plots any return less than 10 feet away on top of an existing floor map (for visual feedback). Three options are supported:

(1) display only the changing ranges

(2) display only changing ranges where an infrared motion detector also indicates motion

(3) power up the camera and enable head tracking of the intruder.

This same function has been integrated with the intrusion detection module, so it is not really needed any more.

PE C 46324 08-07-90 2:50p

The position estimation module has two parts: (1) building a sonar database, and (2) dead reckoning position correction. The database is stored in a separate file from the floor map but each is represented in the "map" data structure.

/**********Utility modules**********/

MAPEDIT C 19071 08-08-90 10:59a

The map editor. The editor allows for the editing of floor map file structures. In addition, fixed sensor coverage areas and doorways can be manipulated.

More doorway and fixed sensor editing features need to be added, e.g. move, add or delete segments to or from the sensor coverage polygons.

LOGGER C 8655 04-09-90 4:13p

This module implements the handling of log files for the intruder module, both logging and replay. An attempt is made at data encapsulation.

MAP C 33547 08-09-90 4:52p

This module contains all the routines for low-level manipulations of the maps. This module is very long and somewhat complete. Only rarely do the individual fields of a map structure need to be manipulated directly.

Future improvements include being able to handle large maps (bigger than 64K) and possibly some way of modeling walls that would be suitable for wall following.

LINKRBRT C 17410 05-08-90 10:45a

This module handles all the packet level manipulations. The module logically sits on top of the serial I/O library, so it is the lowest level of communication with the robot.

PACKET C 27631 07-16-90 1 2:20p

This module contains procedures to decode packets into internal data structures. The module also has a routine "Get_Packet" which handshakes request packets to ROBART with data packets coming back.

IBMIO C 17877 08-09-90 9:44a

This module contains many of the low-level routines for doing I/O operations on the computer, most concerned with keyboard and screen I/O.

IBMRTCLK C 5924 05-09-90 12:49p

This is the IBM realtime clock module, which is used by the intruder module to time events, the packet module for packet time outs, and to download the time to the robot's realtime clock at initial startup.

DEADRECK C 10546 08-09-90 5:28p

This software does the interim dead reckoning while the robot is moving. The module will be removed when the dead reckoning software has been ported to the robot.

LINESTUF C 5354 06-06-90 10:24a

This module contains routines for drawing lines and circles. It is used by any module that needs to draw a line on the map or just calculate all the points along a line between two enpoints.

MISC C 18561 08-09-90 9:24a

Miscellaneous C functions.

OA C 19668 08-07-90 2:28p

This module performs the object mapping and takes care of planning paths around obstacles that occlude ROBART's path.

Future enhancements would incorporate data from the near-infrared proximity sensors. It would also be nice occasionally to back up a few inches before performing an avoidance maneuver.

SCRIPT	C	30969 08-08-90	4:33p

This software interprets all the commands from a script file. The current version is fairly crude, though all commands the robot is capable of executing can be done from a script.

A control structure needs to be added to the language, e.g., "for/while" loops and "if" statements.

TTY	C	7802 08-01-90	8:57a

A simple terminal emulator module, used for talking directly to the Scheduler during debugging.

We need to add a "Capture" feature to allow data capture to RAM or disc, and a "Print" feature that enables hardcopy output.

FIXEDSEN	C	16210 08-07-90	2:21p

This module is similar to map.c, but it handles all the maniuplations for the fixed sensors.

DISPLIST	C	15278 06-06-90	10:19a

The routines in this file are used for creating, displaying and maintaining display lists. Display lists are used primarily by the teleoperated screen and intruder display.

SQRT	C	966 06-06-90	10:45a

This is an integer square root routine needed by the global path planner. It implements the Newton-Raphson iteration.

MINTURNS	C	14392 04-05-90	3:45p

This is one of the routines used for optimizing paths. This one minimizes the number of turns in a path subject to certain constraints.

FIXED	C	4094 08-09-90	8:10a

This module contains functions used by the display list for showing the coverage areas of fixed sensors, and also for reading the state of the sensors.

RECAL	C	16800 08-07-90	3:17p

These functions handle recalibration of the robot. This consists of finding the recalibration position, planning a path to it, executing the recalibration procedure, then interpolating the robot's position from the map and sonar data.

A way of generalizing the recalibration position needs to be implemented. This will also allow multiple recalibration locations rather than just one.

POLYFILL	C	5904 04-05-90	3:58p

This is a routine that will fill a polygon with a given value. Used by the fixed sensor routines to fill in the coverage bit maps.

MAPCUR	C	14857 08-09-90	9:30a

Routines for manipulating the map cross-hairs and for doing some of the rubber-banding functions.

EXECUTE C 19380 08-09-90 8:51a

Path execution routines. These routines will take a path consisting of X-Y coordinates and generate the appropriate set of turn and move commands to execute that path.

SIO C 2152 04-05-90 3:40p

Miscelleneous serial I/O routines.

SPEECH C 7949 04-09-90 5:09p

This is a speech compiler. It takes english words as input and outputs the appropriate codes to the speech computer.

CELLIST C 8786 06-06-90 11:01a

A set of routines for adding and deleting cells from the cell list. The cell list is used by the global path planner.

TELE C 43057 08-08-90 5:02p

This file contains functions for the reflexive-teleoperated screen. By and large, this is just a graphics intensive module rather than a computing module.

A big future enhancement would be to display a scrolling map of obstacles detected by the sonars as they pass by. Perhaps include a mode where the X-Y position on the map is displayed.

GLOBAL C 9911 08-07-90 2:13p

This is the global path planning module. It plans paths that preferentially follow stripe guidepaths rather than just the shortest distance.

There also should be a way to use virtual paths if they are available. Significant work needs to be done to smooth out the paths as they tend to be very jagged.

FILL1 C 3957 06-05-90 1:09p

Contains both graphics fill and flood algorithms used by the teleoperated screen. Needed because the equivalent graphics library routines have bugs.

RECHARGE C 8748 08-07-90 2:10p

Routines for a planning a path to the recharging station and executing the recharging routine.

Future enchancements: (1) allow multiple recharging stations, with a path planned to the closest one, (2) search for a recharging station if none is found in the area, planning paths to other rooms as necessary.

INDISP C 32601 08-09-90 8:12a

Display manipulation routines for the Security Assessment screen.

This should probably be modified to use the display list processor.

ZOOM C 13536 08-09-90 9:46a

Functions for zooming and panning the map.

/**********Include files**********/

ASYNCH	H	260	01-11-89	8:22a
BAUDRATE	H	291	06-01-88	1:58a
DISPLIST	H	1295	01-16-90	10:17a
FUNCKEYS	H	2515	12-28-89	10:22a
LINESTUF	H	662	12-28-89	10:21a
LINKDEFS	H	1053	02-02-89	5:01p
ROBICONS	H	29608	08-07-90	8:36a
PACKET	H	1161	12-28-89	10:21a
QUEUE	H	561	04-11-89	12:16p
RTCLK	H	909	03-30-89	4:51p
SENSOR	H	143	12-28-89	10:21a
SERIAL	H	4109	12-28-89	10:22a
SYSCALL	H	4266	06-01-88	2:01a
VOX1	H	1176	12-28-89	10:22a
VOX2	H	1308	12-28-89	10:22a
PROTOS	H	17786	08-09-90	9:13a
VOX4	H	523	12-28-89	10:22a
VOX3	H	1190	08-08-90	5:05p
IBMIO	H	3566	05-09-90	12:58p
MAPDEFS	H	1764	04-12-90	1:53p
MYTYPES	H	7713	08-02-90	9:41a

/**********Utility files**********/

MAKEFILE 3619 08-09-90 11:41a

Makefile for this Lattice C implementation. The only valid target is "robart".

BLANK MAP 4483 08-01-88 4:41p

A blank map used during demonstrations of the map making process.

NLAB MAP 7886 05-18-90 12:57p

The floor map of the lab space.

F36 MAP 38857 08-08-90 12:05p

Map of the whole fieldhouse.

APPENDIX B

PACKET DEFINITION

PACKET LEVEL

Byte No.	Value	Meaning
0	'!'	Start of Packet
1		Packet Identifier
2		Start of Data
.		.
.		.
N		.
N+1	CR-OxOD	End of Packet

* start of input buffer ($7e00) in Scheduler RAM
start of output buffer ($7f00) in Scheduler RAM

The packet is sent at 1200 bits per second over an RS-232 serial RF link, from the IBM 80386 (Planner) to the robot (Scheduler), and vice versa.

REPORT TYPE DEFINITIONS

Packet Type	Value	Meaning
'1' =	49	Not currently used
'2 =	50	Robot Status Report
'3' =	51	Navigation Sonar Report
'4' =	52	Intrusion Sensor Report
'5' =	53	Displacement Marker Report
'6' =	54	Dead Reckoning Heading
'7' =	55	Collision Sonar Report
'8' =	56	Not currently used
'9' =	57	Move Complete Report

Robot Command Packet Format

Byte #	Name	Interpretation
0	Type Identifier	Packet type = '0'
1	command	Robot command, can either be an action command or a request for sensor data command
2 -> N	Parameters	Parameters which apply to the previous command.

ALLOWABLE COMMANDS

'C' = Charge/Calibrate command

Bytes 2,3—Type action requested:

00—Used to initiate a wall calibration sequence, wherein the robot advances to a wall of known orientation, aligns itself to a normal, and advances slowly until contacting the wall, whereupon it reports ranges to the left and right

RESPONSE: Reports ranges to the left and right in accordance with the following format:

0CXXXXYYYY

where: XXXX is the range to the left
YYYY is the range to the right

Ranges are in hexadecimal format, with a resolution of a tenth of an inch. The routine is used to update the robot's position and orientation. The following error codes apply:

0E01—Excessive initial range error (range > 51.2")
0E02—Blockage error (obstacle detected)
0E03—I/R confirmation error (incorrect response from near-infrared scanners)

01—Used to undock from the recharging station or the recalibration station. Causes the robot to back up the specified distance, at speed '11', whereupon it issues a move complete report, with Stop Code set to '05'.

Bytes 2,3—Distance to move, inches, in hex

RESPONSE: Issues a move complete report, with Stop Code set to '05'.

02—Used to initiate docking procedure with the recharging station.

'D' = Diagnostic command

Used by Host to initiate diagnostic routines.

RESPONSE: Command is echoed back with prefix 0 changed to a 9.

'F' = Follow command

Used to initiate 'Follow' mode, wherein the robot follows a moving target at a specified interval. Future parameters will indicate target range and bearing. Robot exits 'Follow' mode upon receipt of any subsequent command.

'H' = Head Position command (Camera Pan)

Byte 2—the desired head controller action:

0—Assume a discrete head position
1—Center the head
2—Jog the head left
3—Jog the head right

Bytes 3,4—desired head position in hex if byte 1 = '0'
—jogging interval if byte 1 = '2' or '3'

'M' = Move command

The next series of bytes specify: (1) the direction in which to move or pivot, (2) the velocity, and (3) the duration (distance or number of degrees).

Bytes 2,3—Type motion Requested (00 to 07)

00—Move forward
01—Move reverse
02—Pivot left
03—Pivot right
06—Follow guidepath stripe
07—Stop

Bytes 4,5—Requested Velocity (00 to FF)

Byte 4 is the port motor velocity
Byte 5 is the starboard motor velocity

Bytes 6,7—Distance to Move in Inches (00 to FF) or,
Duration of Pivot in Degrees (00 to FF)

Note: This format is consistent with protocol for communication between the Scheduler (CPU #1) and the Drive controller (CPU #4).

RESPONSE: Command is echoed back at completion of path segment, with prefix 0 changed to a 9.

In addition, the following information is added for Mode 6:

Bytes 8, 9, 10, 11—X coordinate in tenths of inches
Bytes 12, 13, 14, 15—Y coordinate in tenths of inches
Bytes 16, 17, 18, 19—current robot heading in degrees/100
Bytes 20, 21, 22, 23—average heading in degrees/100

And for all modes, the last four bytes are as follows:

Bytes N, N-1—battery voltage
Bytes N-1, N-1—stop code

Stop codes are as follows:

00—not currently used
01—normal move complete
02—potential collision detected
03—new command received from Host
04—bumper contact
05—undocking complete

'N' = Navigation parameter command

Used by the Host to set the navigation parameters (X,Y, θ).

Bytes 2,3—Specify the action to be taken:

00—not currently used
01—set navigation parameters

Bytes 4, 5, 6, 7—X coordinate in tenths of inches
Bytes 8, 9, 10, 11—Y coordinate in tenths of inches
Bytes 12, 13, 14, 15—heading in hundredths of degrees

'Q' = Query command

Used by the Host to query RF link status/integrity. Preceded by a packet identifier (byte 0) which indicates addressee of query according to following scheme:

0 - robot
1 - local (host) modem
2 - remote (robot) modem

Addressee responds by echoing back the query command to the host. Host polls the local modem, then the remote modem, and finally the robot to validate link integrity before issuing a 'power up" X command.

'R' = Request command

The next byte is the requested packet type specifying which report packet has been requested.

1—Not currently used
2—Robot Status Report
3—Navigation Sonar Report
4—Intrusion Sensor Report
5—Displacement Marker Report
6—Dead Reckoning Report
7—Collision Sonar Report
8—Not currently used

'S' = Security Mode command

Used to activate the Security Assessment Mode. Deactivated by any subsequent command.

'T' = Time Set command

This command is used to download the time of day from the host computer to the realtime clock onboard the robot at initial startup.

Bytes 2,3 - Hours, in hex
Bytes 4,5 - Minutes, in hex
Bytes 6,7 - Seconds, in hex

'V' = VOX command

This command is used to implement speech routines at the Planner level.

Bytes 2,3—Specify the VOX word list
Bytes 4,5—Specify the word number on that list

This pattern is repeated for all the words in the string, which is then terminated with a carriage return.

'W' = Wander command

This command is used to activate/deactivate "Wander" Mode (Reflexive Teleoperated Control), and to specify the preferred direction for turning in the event an obstacle is detected. It allows the superposition of reflexive collision avoidance on top of remote driving commands entered by the operator while in the Teleoperated Mode.

Bytes 2,3—indicate direction of preferred turn if set:

00—disabled
01—turn left
02—turn right

The actual W Command Parser (W.Parse) can recursively call the Command Parser, so as to read Move Commands and Head Position Commands, for example.

'X' = Execute command

This command is used for miscellaneous On/Off type operations. The next two bytes identify the device or function under control, and the third byte indicates the desired state.

Bytes 2,3—Device

00—System power
01—Siren power
02—Camera power
03—Security suite power
04—SONAR power
05—Speech power
06—Collision array output
07—Speed governing disable
08—Reset Emergency Abort

Byte 4—Action to be taken:

0—Turn off
1—Turn on

Collision Sonar Sensor Report Packet Format

Byte #	Name	Interpretation
0	Type Identifier	Packet type = '1'
1 - N	Sonar Range data	Navigational Sonar Ranges
N	CR	

Robot Status Report Packet Format

Byte #	Name	Interpretation
0	Type Identifier	Packet type = '2'
1,2	Security sensors	Bit0 = Toxic gas Bit1 = Smoke Bit2 = Flooding Bit3 = Bit4 = Bit5 = Bit6 = Bit7 = Access doors
3,4	Internal Temp.	Hex
5,6	External Temp.	Hex
7,8	Battery Voltage	Hex
9,10	Relative Humidity	Hex
11,12	Not used	
13	CR	

Navigational Sonar Sensor Report Packet Format

Byte #	Name	Interpretation
0	Type Identifier	Packet type = '3'
1 – N	Sonar Range data	Navigational Sonar Ranges
N	CR	

Intrusion Sensor Report Packet Format

Byte #	Name	Interpretation
0	Type Identifier	Packet type = '4'
1,2	Security sensors	Bits 0 to 7 represent the status of sensors 1 to 8, respectively. 0 = Not Available and 1 = Online. Bit0 = Infrared motion Bit1 = Microwave motion Bit2 = Ultrasonic motion Bit3 = Optical motion Bit4 = Discriminatory Hearing Bit5 = Vibration Monitor Bit6 = Video Motion Bit7 = (Not Used)
3,4	IR motion value	Bits 0 to 4 specify ON/OFF for each of the 5 IR sensors. Bit0 = IR1 (+43.8 deg) Bit1 = IR2 (+17.2 deg) Bit2 = IR3 (–17.2 deg)

Intrusion Sensor Report Packet Format (Continued)

Byte #	Name	Interpretation
		Bit3 = IR4 (-43.8 deg) Bit4 = IR5 (0.0 deg)
5,6	Microwave motion Detector Value	Bits 0 to 3 give the current sensitivity value in 16 steps. Bit4 gives the alarm condition.
7,8	Head position	Position of head in hex
9,10	Not used	
11,12	Optical motion Detector Value	Bits 0 to 3 specify ON/OFF for each of the 4 optical detectors. Bit0 = OM1 (+135 deg) Bit1 = OM2 (+ 45 deg) Bit2 = OM3 (- 45 deg) Bit3 = OM4 (-135 deg) These angles are relative to the head position.
13,14	Vibration Detector Value	Bit0 specifies ON/OFF alarm condition.
15,16	Discriminatory Hearing Value	Value representing the bearing to disturbance, in degrees from 1 to 180. If the value is 255, there is no bearing solution.
17,18	Discriminatory Hearing Flag	Bits 0 to 2 specify ON/OFF for each of the 3 hearing sensors. Bit 7 is sign of angle above.
19,20	Composite Assessment Value	Value used to compare against alarm threshold for alarm assessment.
21 - N-1	Sonar data	Ranges to any detected motion
N	CR	

Navigational Sonar Plus I/R Report Packet Format

Byte #	Name	Interpretation
0	Type Identifier	Packet type = '5'
1 - N	Sonar Range data	Navigational Sonar Ranges
N, N+1	IR motion value	Bits 0 to 4 specify ON/OFF for each of the 5 IR sensors. Bit0 = IR1 (+43.8 deg) Bit1 = IR2 (+17.2 deg) Bit2 = IR3 (-17.2 deg) Bit3 = IR4 (-43.8 deg) Bit4 = IR5 (0.0 deg)
N+2	CR	

Recharging Status Report Packet Format

Byte #	Name	Interpretation
0	Type Identifier	Packet type = '8'
1,2	Recharging Status	$00 = Initial Acquisition $01 = $02 = $ff = Connected to Charger
3,4	Head Position	Head position in hex (00 to FF)
5,6	Beacon Range	Scaled range in hex
7	CR	

APPENDIX C

PORT ASSIGNMENTS

CPU #1 (Scheduler) Port Assignments

6522-1

ora1 (A Conn)

14 - PAO - (orange)	MO mode select to MMC-02-05 Pin S
4 - PA1 - (red)	M1 mode select to MMC-02-05 Pin T
3 - PA2 - (brown)	M2 mode select to MMC-02-05 Pin U
2 - PA3 - (blue)	CPU Busy Signal from MMC-02-05 Pin V
5 - PA4 - (brown)	CPU #2 power up
6 - PA5 - (orange)	CPU #3 power up
7 - PA6 - (black)	CPU #4 power up
8 - PA7 - (blue)	CPY #5 power up

orb1 (A Conn)

9 - PBO - (black)	CPU #2 data latch (to Pin B MMC-02-2)
10- PB1 - (orange)	CPU #3 data latch (to Pin B MMC-02-3)
11 - PB2 - (grey)	CPU #4 data latch (to Pin B MMC-02-4)
12 - PB3 - (red)	CPU #5 data latch (to Pin 3 MMC-02-5)
13 - PB4 - (red)	interface power up
16 - PB5 - (blue)	RS-232 Clear To Send (CTS)
- PB6 -	not available
15 - PB7 - (grey)	RS-232 Request To Send (RTS)

6522-2

ora2 (AA Conn)

D - PAO -	A/D output bit 0
3 - PA1 -	A/D output bit 1
C - PA2 -	A/D output bit 2
12 - PA3 -	A/D output bit 3
N - PA4 -	A/D output bit 4
11 - PA5 -	A/D output bit 5
M - PA6 -	A/D output bit 6
10 - PA -	A/D output bit 7

orb2 (AA Conn)

L - PB0 -	camera power up
9 - PB1 -	beacon control transmitter
K - PB2 -	unused output
B - PB3 -	unused output
J - PB4 - blue	security sensors power up
7 - PB5 -	
H - PB6 -	
6 - PB7 -	head position A/D flag

6522-3

ora3 (AA Conn)

V	PA0 –	Write Protect Monitor RAM
W	PA1 –	Siren (to pin F on IF#5)
X	PA2 – (grey)	I/R control line PO (LED Array)
18	PA3 – (brown)	I/R control line P1 (LED Array)
19	PA4 – (white)	I/R power up
20	PA5 –	temporary VOREC strobe
17	PA6 – (blue)	256 input data selector read
U	PA7 – (red)	EOC A/D conversion

orb3 (AA Conn)

16	PB0 – (black)	C0 – RS-232 MUX control
T	PB1 – (grey)	C1 – RS-232 MUX control
15	PB2 – (blue)	C2 – RS-232 MUX control
S	PB3 – (orange)	optical motion enable (not used)
Y	PB4 – (brown)	microwave chip enable
21	PB5 – (black)	microwave sensitivity
Z	PB6 –	
22	PB7 –	

6532

ora4 (A Conn)

21	PA0 – (yellow)	AO – 8 line data bus
19	PA1 – (white)	A1 – 8 line data bus
Y	PA2 – (black)	A2 – 8 line data bus
22	PA3 – (yellow)	A3 – 8 line data bus
20	PA4 – (white)	A4 – 8 line data bus
18	PA5 – (black)	A5 – 8 line data bus
W	PA6 – (yellow)	A6 – 8 line data bus
17	PA7 – (white)	A7 – 8 line data bus

orb4 (A Conn)

K	PB0 –
L	PB1 –
M	PB2 –
N	PB3 –
P	PB4 –
T	PB5 –
N/C	PB6 –
N/C	PB7 –

CPU #2 (Head Controller) Port Assignments

IO1 oral

4 - CA2 - (blue)	start A/D conversion
15 - PA0 -	b0 - Speed command
16 - PA1 -	b1 - Speed command
17 - PA2 -	b2 - Speed command
18 - PA3 -	b3 - Speed command
19 - PA4 - (grey)	head motor direction
20 - PA5 - (white)	interface power up
21 - PA6 - (red)	RS-232 Request To Send
22 - PA7 - (black)	RS-232 Clear To Send

IO1 orb1

06 - CB2 - (orange)	beacon XMT control line
07 - PB0 - (brown)	A0 - A/D address
08 - PB1 - (orange)	A1 - A/D address
09 - PB2 - (blue)	A2 - A/D address
10 - PB3 - (grey)	A3 - A/D address
11 - PB4 - (orange)	M0 - mode control lines from SYM
12 - PB5 - (red)	M1 - mode control lines from SYM
13 - PB6 - (brown)	M2 - mode control lines from SYM
14 - PB7 - (red)	EOC - (from A/D)

IO2 ora2

C - CA2 - (blue)	RS-232 Out
B - CA1 - (black)	latches Port A
S - PA0 - (green)	8 bit data bus
T - PA1 - (grey)	8 bit data bus
U - PA2 - (brown)	8 bit data bus
V - PA3 - (black)	8 bit data bus
W - PA4 - (orange)	8 bit data bus
X - PA5 - (red)	8 bit data bus
Y - PA6 - (blue)	8 bit data bus
Z - PA7 - (white)	8 bit data bus

IO2 orb2

H - PB0 - (red/yel)	A/D output C to SYM D
J - PB1 - (blu/yel)	A/D output D to SYM 3
L - PB2 - (blk/blk)	A/D output E to SYM C
K - PB3 - (brn/yel)	A/D output F to SYM 12
N - PB4 - (grn/wht)	A/D output H to SYM N
N - PB5 - (org/blk)	A/D output J to SYM 11
P - PB6 - (yel/yel)	A/D output K to SYM M
R - PB7 - (gry/wht)	A/D output L to SYM 10

MISC

A - 02 - (yellow)	1 MHz clock out

CPU #3 (Sonar Controller) Port Assignments

IO1 ora1

15 - PA0 -		
16 - PA1 -		
17 - PA2 -		
18 - PA3 -		XLOG3 transmit flag
19 - PA4 -		MFLOG3 receive flag
20 - PA5 -		VSW3 transmit trigger
21 - PA6 -		XLOG2 transmit flag
22 - PA7 -		MFLOG2 receive flag

IO1 orb1

07 - PB0 -	(black)	A0 - Sonar select address
08 - PB1 -	(red)	A1 - Sonar select address
09 - PB2 -	(brown)	A2 - Sonar select address
10 - PB3 -	(orange)	A3 - Sonar select address
11 - PB4 -	(white)	interface power up (to IF #11 pin H)
12 - PB5 -	(blue)	VSW1&2 transmit trigger
13 - PB6 -	(black)	XLOG1 transmit flag
14 - PB7 -	(orange)	MFLOG1 receive flag

IO2 ora2

B - CA1 -	latches Port A (jumper 10 to 22 on AD2)
S - PA0 -	8 bit data bus from SYM
T - PA1 -	8 bit data bus from SYM
U - PA2 -	8 bit data bus from SYM
V - PA3 -	8 bit data bus from SYM
W - PA4 -	8 bit data bus from SYM
X - PA5 -	8 bit data bus from SYM
Y - PA6 -	8 bit data bus from SYM
Z - PA7 -	8 bit data bus from SYM

IO2 orb2

H - PB0 -	(grey)	Request To Send (RTS) handshake
J - PB1 -		
L - PB2 -		
K - PB3 -		
D - PB4 -		RS-232 (TTY) out
N - PB5 -		
P - PB6 -		
R - PB7 -	(blue)	Clear to Send (CTS) handshake

CPU #4 (Drive Controller) Port Assignments

IO1 ora1

15 - PA0 - (blue)	d0 - CDP 1878 data
16 - PA1 - (red)	d1 - CDP 1878 data
17 - PA2 - (brown)	d2 - CDP 1878 data
18 - PA3 - (grey)	d3 - CDP 1878 data
19 - PA4 - (green)	d0 - CDP 1878 data
20 - PA5 - (black)	d1 - CDP 1878 data
21 - PA6 - (orange)	d2 - CDP 1878 data
22 - PA7 - (grey)	d3 - CDP 1878 data

IO1 orb1

07 - PB0 - (grey)	A0 - CDP 1878 address
08 - PB1 - (black)	A1 - CDP 1878 address
09 - PB2 - (white)	A2 - CDP 1878 address
10 - PB3 - (yellow)	write strobe
11 - PB4 - (white)	Interface 10 (drive) power up - IF 10/13
12 - PB5 -	port motor direction
13 - PB6 - (grey)	port motor tach input - IF 10/8
14 - PB7 -	stbd motor direction

IO2 ora2

B - CA1 -	Port A latch (jumper 10 to 22 on AD2)
S - PA0 -	SYM command (8 bit bus)
T - PA1 -	SYM command (8 bit bus)
U - PA2 -	SYM command (8 bit bus)
V - PA3 -	SYM command (8 bit bus)
W - PA4 -	SYM command (8 bit bus)
X - PA5 -	SYM command (8 bit bus)
Y - PA6 -	SYM command (8 bit bus)
Z - PA7 -	SYM command (8 bit bus)

IO2 orb2

H - PB0 - (orange)	M0 - mode control
J - PB1 - (red)	M1 - mode control
L - PB2 - (brown)	M2 - mode control
K - PB3 -	
M - PB4 -	
N - PB5 -	
P - PB6 - (yellow)	stbd motor tach input - IF 10/9
R - PB7 -	

MISC

A - O2 - (blue)	1 mHz clock out

CPU #5 (Speech Controller) Port Assignments

IO1 ora1

3	- CA1 -		Port latch from SYM A-12
15	- PA0 -	(green)	A0
16	- PA1 -	(grey)	A1
17	- PA2 -	(brown)	A2
18	- PA3 -	(black)	A3
19	- PA4 -	(orange)	A4
20	- PA5 -	(red)	A5
21	- PA6 -	(blue)	A6
22	- PA7 -	(white)	A7

IO1 orb1

07	- PB0 -	(orange)	d0 to SS-49
08	- PB1 -	(black)	d1 to SS-48
09	- PB2 -	(lt. blue)	d2 to SS-47
10	- PB3 -	(red)	d3 to SS-46
11	- PB4 -	(grey)	d4 to SS-45
12	- PB5 -	(white)	d5 to SS-44
13	- PB6 -	(yellow)	d6 to SS-43
14	- PB7 -	(orange)	d7 to SS-42

IO2 ora2

S	- PA0 -	(orange)	M0 - mode select control lines from SYM
T	- PA1 -	(red)	M1 - mode select control lines from SYM
U	- PA2 -	(brown)	M2 - mode select control lines from SYM
V	- PA3 -	(blue)	CPU busy signal out to SYM
W	- PA4 -		
X	- PA5 -		
Y	- PA6 -		
Z	- PA7 -		

IO2 orb2

H	- PB0 -	(grey)	Speech synthesizer power up
J	- PB1 -		
L	- PB2 -	(black)	synthesizer chip select to SS-41
K	- PB3 -	(lt. blue)	synthesizer trigger (R/W) to SS-18
M	- PB4 -	(orange)	A1 speech address to SS-3
N	- PB5 -	(red)	A2 speech address to SS-4
P	- PB6 -	(black)	A3 speech address to SS-5
R	- PB6 -	(yellow)	synthesizer busy to SS-20

MISC

CPU #9 (Communications Controller) Port Assignments

IO1 ora1

3 - CA1 -
15 - PA0 -
16 - PA1 -
17 - PA2 -
18 - PA3 -
19 - PA4 -
20 - PA5 -
21 - PA6 -
22 - PA7 -

IO1 orb1

07 - PB0 -
08 - PB1 -
09 - PB2 -
10 - PB3 -
11 - PB4 -
12 - PB5 -
13 - PB6 -
14 - PB7 -

IO2 ora2

S - PA0 -
T - PA1 -
U - PA2 -
V - PA3 -
W - PA4 -
X - PA5 -
Y - PA6 -
Z - PA7 -

IO2 orb2

H - PB0 -
J - PB1 -
L - PB2 -
K - PB3 -
M - PB4 -
N - PB5 -
P - PB6 -
R - PB6 -

MISC

CPU #10 (Acoustical Processor) Port Assignments

IO1 ora1

3 - CA1 -
15 - PA0 -
16 - PA1 -
17 - PA2 -
18 - PA3 -
19 - PA4 -
20 - PA5 -
21 - PA6 -
22 - PA7 -

IO1 orb1

07 - PB0 -
08 - PB1 -
09 - PB2 -
10 - PB3 -
11 - PB4 -
12 - PB5 -
13 - PB6 -
14 - PB7 -

I02 ora2

S - PA0 -	(purple)	Acoustical element number 1
T - PA1 -	(orange)	Acoustical element number 2
U - PA2 -	(blue)	Acoustical element number 3
V - PA3 -		
W - PA4 -		
X - PA5 -		
Y - PA6 -		
Z - PA7 -		

IO2 orb2

H - PB0 -	(orange)	RTS
J - PB1 -		
L - PB2 -		
K - PB3 -		
D - PB4 -	(brown)	RS-232 out
N - PB5 -		
P - PB6 -		
R - PB6 -	(blue)	CTS

MISC

APPENDIX D

BUS ASSIGNMENTS

44 Pin ZIF Breakout (board #11)

Pin	Color	Function
1	red	+12V
2	red	+12V
3	grey	integrity C
4	black	integrity F
5	red	12-volt battery level sense
6	red	integrity H
7	red	battery voltage A/D pickoff
8	brown	drive enable
9	blue	I/R square wave output
10	green	integrity E
11	brown	integrity B
12	blue	Receiver drive disable
13	brown	microwave chip enable
14	black	microwave sensitivity select
15	orange	motion detector enable
16	brown	room temperature output
17	purple	selector 6 output
18	red	integrity A
19	blue	selector 5 output
20	blue	integrity D
21	grey	card #11 integrity
22	orange	beacon xmtr activate
A	red	+12 volts to I/R range receiver
B	orange	selector 7(head) output
C	grey	photocell output
D	grey	photocell output
E	green	A0
F	grey	A1
H	brown	A2
J	black	A3
K	red	selector 4 output
L	grey	LED array control line P0 (F4)
M	brown	LED array control line P1 (F5)
N	blue	hearing sensor #1
P	blue	selector 8 output (rear doors)
R	orange	hearing sensor #2
S	purple	hearing sensor #3
T	yellow	camera +12 volts
U	yellow	analog signal head position pot

V	red	+12 volts (smoke detector)
W	yellow	+5V head position pot
X	white	head motor energize
Y	green	head motor energize
Z	red	+12V (security sensors)

Ribbon Cable #A (Head Position control)

Pin #	Color	Function
1 –		
2 –	+12V	(motion) > no longer used
3 –	green	energize motor
4 –	white	energize motor
5 –	green	gnd to sensing pot
6 –	yellow	+5v to sensing pot
7 –	blue	analog signal from sensing pot
8 –	red	integrity for cable A (to pin 12 IF #11)
9 –		
10 –		
11 –		
12 –		
13 –	+12V	(motor) > no longer used
14 –	+12V	(motor) > no longer used

Ribbon Cable #B (Video Camera, Acoustical Array)

Pin #	Color	Function
1	green	GND
2	yellow	+12V (hearing array)
3	green	Camera GND
4	green	Camera GND
5	green	Camera GND
6	red	+12 volts (camera)
7	red	+12 volts (camera)
8	red	+12 volts (camera)
9	red	+12 volts (camera)
10	blue	hearing sensor #1
11	orange	hearing sensor #2
12	purple	hearing sensor #3
13	brown	integrity (to pin 11 on IF #11)
14	grey	receiver output "kill"

Ribbon Cable #C Head Interface Board

Pin #	Color	Function
1	brown	microwave chip enable (grey on MW–4)
2	black	microwave sensitivity select (blue on MW–K)

3		+12 volts (interface board #15)
4	orange	A0
5	blue	A1
6	purple	A2
7	grey	A3
8		selector 7 output
9		+12 volts (I/R range unit receiver)
10	blue	+12 volts (optical motion detectors)
11	grey	+12 volts (microwave motion detector)
12	black	integrity (to pin 3 on IF #11)
13		photocell output
14		photocell output

Ribbon Cable #D — Infrared Motion Detector MUX

Pin #	Color	Function
1		
2		A0 lower
3		A1 nibble of
4		A2 data bus
5		A3
6	blue	integrity sense D (to pin 20 I/F #11)
7		selector output #5 (onboard) (to pin, IF #11)
8		floor sensors interrupt
9		+12V
10		selector 4 output
11		selector 4 interrupt (I/R pres del)
12		integrity check I (sel 5)
13		selector 6 output (to pin 17, IF #11)
14		selector 6 interrupt (bumpers)
15		integrity check J (sel 6)
16		+12 volts (I/R motion detectors)

Ribbon Cable #E

Pin #	Color	Function
1	blue	I/R square wave drive
2		integrity sense H (to Pin H-14)
3		A0
4		A1
5		A2
6		A3
7		room temp sensor output
8		+12 volts (interface bus)
9		+12 volts (smoke detector)
10		
11		

12	blue	selector 8 output (rear doors)
13	green	integrity sense E (to pin 10 IF #11)
14		drive disable from rear panel 'exit' pushbutton (pin 12, IF #11)
15		drive enable from rear panel (to pin 8, IF #11)
16		

Ribbon Cable #F

Pin #	Color	Function
1		
2		
3	blue	I/R square wave drive (IF #11 Pin 9)
4	grey	I/R LED Array control Line P0 (IF #11 Pin L)
5	brown	I/R LED Array control Line P1 (IF #11 Pin L)
6		battery voltage A/D pickoff
7		+12 volts battery level sense
8		
9		
10		
11		
12		
13	black	integrity sense F (to pin on IF #11)

Ribbon Cable #G 16 Pin ZIF Drive Wheel Control

Pin #	Color	Function
1	brown	starboard forward control line
2	siren	enable
3	orange	starboard reverse control line
4	purple	GND
5		
6	orange	drive safety switch verify
7		starboard breaker switch verify
8		+12V drive interface power
9	blue	starboard velocity channel A
10	orange	port velocity channel A
11		port breaker switch verify
12	black	port velocity channel B
13	red	port reverse control line
14	brown	stbd velocity channel B
15	blue	port forward control line
16		integrity sense

Ribbon Cable #H (Outer Skin Connect)

Pin #	Color	Function
1		
2		
3		
4		
5		selector 8 output (rear door)
6		
7		room temperature sensor output
8		+12 volts
9	blue	I/R square wave drive
10		A3
11		A2
12		A1
13		A0
14		integrity check H

Ribbon Cable #I Selector 4

Pin #	Color	Function
1		infrared motion detector #5 output
2		A0
3		A1 Lower nibble of 8 line
4		A2 address
5		A3
6		selector 6 output (bumpers in base)
7		floor sensors output
8		selector 6 interrupts (bumpers floor sensors)
9		integrity check J
10		integrity check I
11		selector 4 output (onboard)
12		selector 4 interrupt
13		
14		+12V to selector 5
15		+12V to selector 6
16		

Ribbon Cable #J Selector 6

Pin #	Color	Function
1		A0
2		A1 lower nibble of 8 line
3		A2 address
4		A3
5		selector 6 output
6		floor sensors interrupt

7		bumper interrupt out
8		integrity check J
9		
10		
11		
12		
13		
14		
15		aux power verify to selector 4
16		+12V

T-connector (soldertail to PC Board) Local Sym

Pin #	Color	Function
1		GND
2		RS 232 in
3		RS 232 out
4		
5		
6		
7		
8		
9		
10		
11		
12		RS 232 (20 MA TTY out)
13		
14		
15		
16		
17		
18	grey	audio in
19		audio GND
20	blue	address decode 11, 12, 13, 14 (bank switch)
21	brown	audio out (hi)
22		reset
23		
24		K – U21 Chip select (bank switch)
25	audio	GND

APPENDIX E

SELECTOR ASSIGNMENTS

Selector	Board	Function
0	IF #2	IF Selector (Circuit board integrity)
1	IF #2	connector integrity
2	IF #6	switch Position
3	IF #6	Circuit Breakers
4A	IF #	Near I/R Proximity Detectors (in shell)
4B	IF #	Near I/R Proximity Detectors (in base)
5	IF #	I/R motion detectors (on I/R motion)
6	IF #	Tactile Bumpers, Floor Sensors (in base)
7	IF #15	Near I/R Programmable/Optical motion M Wave
8A		Left Rear Door (4051)
8B		Right Rear Door (4051)
9		
10		

SELECTOR 0 -- Interface Board Integrity

INPUT	COLOR	FUNCTION
0		GND
1		board 1
2		GND
3		board 3
4		board 4
5		board 5
6		board 6
7		board 7
8		board 8
9		board 9
10		board 10
11		board 11
12		
13		
14		
15		

SELECTOR 1 -- Ribbon Cable Integrity

INPUT	COLOR	FUNCTION
0	red	connector A
1	brown	connector B
2	grey	connector C
3	blue	connector D
4	green	connector E
5	black	connector F
6	grey	connector G
7	red	connector H
8		
9		
10		
11		
12		
13		
14		
15		

SELECTOR 2 -- Switch Position Verify

LOCATED INPUT	COLOR	FUNCTION
0		GND
1	brown	switch 1
2	grey	switch 2
3	blue	switch 3
4	black	switch 4
5	blue	switch 5
6		
7		
8		
9		
10		
11		
12		
13	blue	Drive motion Verify
14	orange	RS-232 Verify
15	orange	Emergency Stop Verify

SELECTOR 3 -- Breaker Switch Verify

LOCATED IN CARD CAGE ON IF-

INPUT	COLOR	FUNCTION
0	brown	Starboard Motor
1	green	Port Motor
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13	blue	Hearing sensor #1
14	orange	Hearing sensor #2
15	purple	Hearing sensor #3

SELECTOR 4 -- Near I/R Proximity Detectors

LOCATED IN SHELL SERVICED BY RIBBON CABLE I

INPUT	COLOR	FUNCTION
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

SELECTOR 5 -- Infrared Motion Detectors

LOCATED IN SHELL ON I/R ARRAY SERVICED BY RIBBON CABLE D

INPUT	COLOR	FUNCTION
0	orange	I/R motion Detector
1	blue	I/R motion Detector
2	grey	I/R motion Detector
3	brown	I/R motion Detector
4		Interrupt Bus for 0-3
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

SELECTOR 6 -- Tactile Bumpers/Floor Sensors

LOCATED IN BASE SERVICED BY RIBBON CABLE J

INPUT	COLOR	FUNCTION
0	dark blue	Front Panel Left Bumper
1	white	Front Panel Right Bumper
2	orange	Left Side Panel Front Bumper
3	yellow	Right Side Panel Front Bumper
4	brown	Rear Panel Left Bumper
5	black	Rear Panel Right Bumper
6	grey	Left Side Panel Rear Bumper
7	light blue	Right Side Panel Rear Bumper
8	white	Forward Right Banner (base)
9	white	Forward Left Banner (base)
10		Floor Sensor
11		Floor Sensor
12		Floor Sensor
13		Floor Sensor
14		Floor Sensor
15		Floor Sensor

SELECTOR 7 -- Interface Board #12

LOCATED ON HEAD SERVICED BY RIBBON CABLE C

INPUT	COLOR	FUNCTION
0		
1		Optical motion #1
2		Optical motion #2
3		Optical motion #3
4		Optical motion #4
5		
6		
7		
8		Microwave motion detector
9		
10		
11		
12	grey	I/R LED #1 to Pin L
13	brown	I/R LED #2 to Pin K
14	orange	I/R LED #3 to Pin 10
15	blue	I/R LED #4 to Pin 9

SELECTOR 8 -- Miscellaneous

LOCATED ON REAR DOOR SERVICED BY RIBBON CABLES E & H

Selector 8A (left door)

INPUT	COLOR	FUNCTION
1		
2		
3		
4		
5	green	"exit" button
6		
7		

Selector 8B (right door)

INPUT	COLOR	FUNCTION
8	green	smoke detectors
9		
10	black	access door
11	green	toxic gas
12		
13		
14	black	I/R (red)
15	purple	I/R (yellow)

APPENDIX F

SYNTHESIZER VOCABULARY WORD LISTINGS

VOX 1 Word List

ZERO	00	Q	30	IS	60
ONE	01	R	31	IT	61
TWO	02	S	32	KILO	62
THREE	03	T	33	LEFT	63
FOUR	04	U	34	LESS	64
FIVE	05	V	35	LESSER	65
SIX	06	W	36	LIMIT	66
SEVEN	07	X	37	LOW	67
EIGHT	08	Y	38	LOWER	68
NINE	09	Z	39	MARK	69
TEN	0A	AGAIN	3A	METER	6A
ELEVEN	0B	AMPERE	3B	MILE	6B
TWELVE	0C	AND	3C	MILLI	6C
THIRTEEN	0D	AT	3D	MINUS	6D
FOURTEEN	0E	CANCEL	3E	MINUTE	6E
FIFTEEN	0F	CASE	3F	NEAR	6F
SIXTEEN	10	CENT	40	NUMBER	70
SEVENTEEN	11	400HERTZ TONE	41	OF	71
EIGHTEEN	12	80HERTZ TONE	42	OFF	72
NINETEEN	13	20MS SILENCE	43	ON	73
TWENTY	14	40MS SILENCE	44	OUT	74
THIRTY	15	80MS SILENCE	45	OVER	75
FORTY	16	160MS SILENCE	46	PARENTHESIS	76
FIFTY	17	320MS SILENCE	47	PERCENT	77
SIXTY	18	CENTI	48	PLEASE	78
SEVENTY	19	CHECK	49	PLUS	79
EIGHTY	1A	COMMA	4A	POINT	7A
NINETY	1B	CONTROL	4B	POUND	7B
HUNDRED	1C	DANGER	4C	PULSES	7C
THOUSAND	1D	DEGREE	4D	RATE	7D
MILLION	1E	DOLLAR	4E	RE	7E
ZERO	1F	DOWN	4F	READY	7F
A	20	EQUAL	50	RIGHT	80
B	21	ERROR	51	SS	81
C	22	FEET	52	SECOND	82
D	23	FLOW	53	SET	83
E	24	FUEL	54	SPACE	84
F	25	GALLON	55	SPEED	85
G	26	GO	56	STAR	86
H	27	GRAM	57	START	87
I	28	GREAT	58	STOP	88
J	29	GREATER	59	THAN	89
K	2A	HAVE	5A	THE	8A
L	2B	HIGH	5B	TIME	8B
M	2C	HIGHER	5C	TRY	8C
N	2D	HOUR	5D	UP	8D
O	2E	IN	5E	VOLT	8E
P	2F	INCHES	5F	WEIGHT	8F

VOX 2 Word List

ABORT	00	FARAD	2C	PER	58
ADD	01	FAST	2D	PICO	59
ADJUST	02	FASTER	2E	PLACE	5A
ALARM	03	FIFTH	2F	PRESS	5B
ALERT	04	FIRE	30	PRESSURE	5C
ALL	05	FIRST	31	QUARTER	5D
ASK	06	FLOOR	32	RANGE	5E
ASSISTANCE	07	FORWARD	33	REACH	5F
ATTENTION	08	FROM	34	RECEIVE	60
BRAKE	09	GAS	35	RECORD	61
BUTTON	09	GET	36	REPLACE	62
BUY	0A	GOING	37	REVERSE	63
CALL	0B	HALF	38	ROOM	64
CAUTION	0C	HELLO	39	SAFE	65
CHANGE	0E	HELP	3A	SECURE	66
CIRCUIT	0F	HERTZ	3B	SELECT	67
CLEAR	10	HOLD	3C	SEND	68
CLOSE	11	INCORRECT	3D	SERVICE	69
COMPLETE	12	INCREASE	3E	SIDE	6A
CONNECT	13	INTRUDER	3F	SLOW	6B
CONTINUE	14	JUST	40	SLOWER	6C
COPY	15	KEY	41	SMOKE	6D
CORRECT	16	LEVEL	42	SOUTH	6E
DATE	17	LOAD	43	STATION	6F
DAY	18	LOCK	44	SWITCH	70
DECREASE	19	MEG	45	SYSTEM	71
DEPOSIT	1A	MEGA	46	TEST	72
DIAL	1B	MICRO	47	TH-	73
DIVIDE	1C	MORE	48	THANK	74
DOOR	1D	MOVE	49	THIRD	75
EAST	1E	NANO	4A	THIS	76
ED (1)	1F	NEED	4B	TOTAL	77
ED (2)	20	NEXT	4C	TURN	78
ED (3)	21	NO	4D	USE	79
ED (4)	22	NORMAL	4E	-UTH	7A
EMERGENCY	23	NORTH	4F	WAITING	7B
END	24	NOT	50	WARNING	7C
ENTER	25	NOTICE	51	WATER	7D
ENTRY	26	OHMS	52	WEST	7E
ER	27	ONWARD	53	WHICH	7F
EVACUATE	28	OPEN	54	WINDOW	80
EXIT	29	OPERATOR	55	YES	81
FAIL	2A	OR	56	ZONE	82
FAILURE	2B	PASS	57		

VOX 3 Word List

-ING	01	FIRE	25	STAIR	49
A-	02	HERE	26	STATUS	4A
ACTIVATE	03	IF	27	STORM	4B
ACTIVE	04	INACTIVE	28	SUPERVISORY	4C
AM (TIME)	05	INSERT	29	TARGET S	4D
AN	06	INTERFACE	2A	TEMPERATURE	4E
AVAILABLE	07	INVALID	2B	TERMINATE	4F
AVERAGE	08	KEYPAD	2C	THANK-YOU	50
BAROMETRIC	09	LINK	2D	THEE-	51
BATTERY	0A	MODULE	2E	THERE	52
BEFORE	0B	MONITOR	2F	TODAY	53
BETWEEN	0C	MOVE	30	TONE	54
BLACK	0D	MY	31	TONE (800)	55
BUILDING	0E	NIGHT	32	TONE (5000)	56
BUSY	OF	O/CLOCK	33	TOUCH	57
CELSIUS	10	OPTICAL M	34	TRACKING S	58
CHANNEL	11	PM (TIME)	35	TRUNK	59
CODE	12	POWER	36	UNABLE	5A
COLD	13	PROGRAM	37	UNATTENDED	5B
COMMAND	14	PUSH	38	UNKNOWN	5C
COMMON	15	PUT	39	USE	5D
COMMUNICATION	16	RECEIVER	3A	VOLTAGE	5E
CONDITION	17	RED	3B	WAIT	5F
CONFIGURATION	18	REMOVE	3C	WAKE-UP	60
CONVERTER	19	REPAIR	3D	WEEK	61
COUNT	1A	REPEAT	3E	WELCOME	62
DATA	1B	RESTORE	3F	WHAT	63
DC	1C	RETURN	40	WORKING	64
DEFAULT	1D	ROUTE	41	YOUR	65
DEMONSTRATION	1E	RUNNNING	42	ABLE	66
DID	1F	SECURITY	43	ACKNOWLEDGE	67
DISABLE	20	SENSOR	44	AWAY	68
ENABLE	21	SEQUENCE	45	BACK	69
EXAMINE	22	SHORT	46	HIT	6A
EXTREME	23	SIGHT	47	LIGHT	6B
FAR	24	STAIR	48	LISTEN	6C

VOX 4 Word List

ALL.CKT	00	DISABLD	27
ASSIST	01	ACTVATD	28
CPU.UP	02	EMR.ACT	29
EX.TIME	03	GREET	2A
FIV.SEC	04	OK	2B
IT.IS	05	REC.OK	2C
LD.TIME	06	TST.STP	2D
LOCK	07	UN.TST	2E
MARK	08	SEC.REC	2F
NOT.NOR	09	RED.UNA	30
SET.TIM	0A	REM.COM	31
SYS.CHK	0B	NOT.ACK	32
SHT.DWN	0C	ACK	33
T.MINUS	0D	PRS.BUT	34
TEST.PT	0F	THS.TIM	35
TH.TIME	10	FAILURE	36
VOX.ON	11	BEEP2	37
VOX.OFF	12	BONG3	38
WAITING	13	CODE	39
SUPER	14	WAT.COM	3A
IF.CKT	15	REP.COM	3B
CONN	16	YES.NO	3C
SWITCH	17	MOD.NUM	3D
CKT.BRK	18	ENT.NUM	3E
UNABLE	19		
BAROM	1A		
RM.TMP	1B		
SYS.TMP	1C		
RS232	1D		
BAT.LOW	1E		
REL.HUM	1F		
DOOR	20		
BAT.VOL	21		
OPT.TRK	22		
DEM.PRO	23		
HIT.RET	24		
UN.MOV	25		
POW.DIS	26		

APPENDIX G

PATENTS

Patents Issued:

H. R. Everett, C. S. Wright
Continuous Tactile Bumper
US Patent # 4,596,412 Navy Case # 68615

H. R. Everett
Programmable Near-Infrared Ranging System
US Patent # 4,851,661 Navy Case # 70153

H.R. Everett
Dual-Element Optical Motion Detector
US Patent # 4,902,887 Navy Case # 71816

H. R. Everett, G. A. Gilbreath
Intelligent Security Assessment System
US Patent # 4,857,912 Navy Case # 71393

H.R. Everett
Hybrid Pneumatic/Hydraulic Actuator
US Patent # To be announced Navy Case # 70227

Applications Submitted:

H.R. Everett
Automatic Recharging System
Navy Case # 70154

H.R. Everett
Reconfigurable Video Line Digitizer
Navy Case # 72498

H.R. Everett, G.A. Gilbreath, R.T. Laird
Hybrid Navigational Scheme for Automated Guided Vehicles
Navy Case # 72770

H.R. Everett, G.A. Gilbreath
Reflexive Teleoperated Control Scheme for Remote Platforms
Navy Case # 72949

Invention Disclosures Submitted:

H.R. Everett, G.A. Gilbreath
Intelligent Data Fusion Scheme for Fixed and Mobile Security Sensors
Navy Case # 72775

H.R. Everett, J. M. Nieusma
Velocity Feedback System for Remotely Operated Vehicles
Navy Case # 73322

H.R. Everett
Doorway Transit Navigational Referencing System
Navy Case # NOSC-835

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2 REPORT DATE	3 REPORT TYPE AND DATES COVERED
	June 1990	Interim: Sep 89 to Jun 90

4. TITLE AND SUBTITLE

MODELING THE ENVIRONMENT OF A MOBILE SECURITY ROBOT

5 FUNDING NUMBERS

PR: ZE50
WU: DN307432
PE: 62936N

6 AUTHOR(S)

H. R. Everett, G. A. Gilbreath, T. Tran, J. M. Nieusma

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Ocean Systems Center
San Diego, CA 92152–5000

8 PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Technology
Independent & Exploratory Development
Washington, DC 20360

10 SPONSORING/MONITORING AGENCY REPORT NUMBER

NOSC TD 1835

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

ROBART II is a battery-powered autonomous robot being used by the Naval Ocean Systems Center (NOSC) in San Diego, CA, as a testbed in research that seeks to provide a multisensor detection, verification, and intelligent assessment capability for a mobile security robot. The intent is to produce a robust automated system exhibiting a high probability of detection, with the ability to distinguish between actual and nuisance alarms, and capable of operation within areas already protected by fixed-installation motion-detection sensors.

An architecture of 13 distributed microprocessors onboard the robot makes possible advanced control strategies and realtime data acquisition. Higher level tasks (map generation, path planning, position estimation, obstacle avoidance, and statistical security assessment) are addressed by a Planner (currently a remote 80386-based desktop computer). Numerous sensors are incorporated into the system to yield appropriate information for use in position estimation, collision avoidance, navigational planning, and assessing terrain traversibility.

Although originally implemented as a purely autonomous vehicle with sophisticated world-modeling capabilities, recent efforts have added an innovative form of remote (teleoperated) control to the system, allowing for manual direction of the robot's motion. This *reflexive teleoperation* frees the operator from the lower level concerns associated with direct teleoperation: speed of the vehicle and vehicle direction are servo-controlled by an onboard processor as a function of the surrounding environment in response to local sensor inputs, but under the high-level supervisory control of the remote operator.

The robot is also equipped with a multitude of sensors for environmental awareness in support of its role as an intelligent sentry. These sensors monitor both system and room temperature, relative humidity, barometric pressure, ambient light and noise levels, toxic gas, smoke, and fire. Intrusion detection is addressed through the use of five passive, true-infrared, body-heat detectors; four passive, optical motion detectors; ultrasonic motion detectors; microwave motion detectors; video motion detection, vibration monitoring, and discriminatory hearing. The realtime security assessment software computes a composite threat analysis by summing the weighted scores of alarmed sensors within a given zone. If the zone composite threat exceeds a dynamically computed threshold, a true alarm condition exists.

14. SUBJECT TERMS

robotic systems
optical proximity sensors
sonar
path planning
collision avoidance
sensors
security
sentry

15 NUMBER OF PAGES

170

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20 LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT